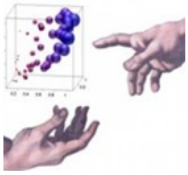


RAPPORT DE STAGE
15 AVRIL AU 21 JUIN 2013

Alexandra GARCIN

Unité Biostatistique et Processus Spatiaux
Institut National de la Recherche Agronomique
Avignon, France
http ://www.biosp.org

Département Statistique et Informatique Décisionnelle
Institut Universitaire de Technologie
Avignon, France
http ://www.iut.univ-avignon.fr

Remerciements

Je tiens à remercier le Directeur du laboratoire BioSP, M. Etienne Klein, pour m'avoir accueillie dans ses locaux. Tout a été mis en place pour que je me sente le plus à l'aise possible et je ne pense pas que ce soit le cas dans toutes les entreprises qui accueillent des stagiaires.
Ensuite j'aimerais remercier Mme Gabriel qui a toujours répondu présente lorsque j'avais des questions à lui poser.

Mais en particulier, j'aimerais remercier mon tuteur, M. Allard. Ce fut un plaisir de travailler avec lui. Il a fait preuve d'une grande patience envers moi. Il a su me pousser au bout de mes limites et a toujours voulu que je fasse du mieux que je pouvais.

Ce stage fut une expérience très enrichissante et je suis fière d'avoir pu le faire à l'INRA.

Sommaire

1 Introduction

Afin de valider le Diplôme Universitaire de Techonologie Statistiques et Informatique Décisionnelle, les étudiants doivent effectuer un stage de dix semaines en entreprise.

Pour ma part, j'ai obtenu un stage au laboratoire Biostatistique et Processus Spatiaux (BioSP) à l'Institut National de la Recherche Agronomique (INRA) du centre Avignon. Je l'ai effectué sous la tutelle de Monsieur Denis Allard, directeur de recherche à l'INRA. La mission qui m'a été confiée était de tester le générateur WACSgen créé par M. Allard.

Mon rapport de stage se fera en deux parties. Dans un premier temps je vais présenter mon organisme d'accueil, le projet sur lequel j'ai travaillé ainsi que la mission qui m'a été confiée. Puis j'exposerai mes méthodes de travail et les résultats que j'ai obtenu.

2 Contexte du stage

2.1 INRA, un Institut de Recherche

De nos jours, la recherche en agronomie doit répondre à des enjeux majeurs, dans un contexte climatique, démographique et énergétique complexe. L'INRA produit des connaissances scientifiques dans les domaines de l'alimentation, de l'agriculture et de l'environnement et est implanté dans 18 centres en région parisienne et en régions. L'Institut comprend plus de deux cents unités de recherche et plateformes expérimentales.

Les recherches sont conduites au sein de 13 départements scientifiques :

- Alimentation humaine : fournit les éléments scientifiques permettant d'améliorer la santé et le bien être en encourageant le développement d'aliments mieux adaptés à l'Homme,
- Biologie et amélioration des plantes : contribue au fait que l'agriculture produise plus et en cohérence avec les besoins,
- Caractérisation et élaboration des produits issus de l'agriculture : s'intéresse aux transformations des matières premières végétales, animales, ou des déchets,
- Ecologie des forêts, prairies et milieux aquatiques : s'occupe de gérer durablement, conserver et restaurer les écosystèmes forestiers, prairiaux et aquatiques ainsi que les ressources physiques et biologiques qui en dépendent,
- Environnement et agronomie : acquiert des connaissances génériques et finalisées pour gérer les espaces cultivés,
- Génétique animale : contribue au progrès des connaissances en génétique et en biologie ; développe les outils d'amélioration génétique des populations animales d'élevage,
- Mathématiques et Informatiques appliquées : produit des connaissances génériques et finalisées de mise au point de méthodes, d'outils et de savoir-faire, dans les domaines des mathématiques, des statistiques et de l'inteligence artificielle,
- Microbiologie et chaîne alimentaire : appréhende le fonctionnement des microorganismes et des écosystèmes microbiens pour mieux les utiliser, les maîtriser ou les combattre,
- Physiologie animale et systèmes d'élevage : conçoit des système d'élevage durables,
- Santé animale : prévient et contrôle les maladies des animaux d'élevage, éclaire la décision publique dans le domaine de la santé publique vétérinaire,
- Santé des plantes et environnement : appréhende le fonctionnement des écosystèmes agricoles,
- Sciences pour l'action et le développement : développe une agro-écologie et contribue à l'innovation,
- Sciences sociales, agriculture et alimentation, espace et environnement : appréhende le fonctionnement et les évolutions économiques et sociales de l'agriculture, des industries agroalimentaires et de l'alimentation.

Mon stage se déroule au sein du département Mathématiques et Informatiques Appliquées, dans le centre de recherche d'Avignon-Montfavet.

Je travaille dans le laboratoire BioSP. Ce laboratoire conduit des recherches en statistique et en modélisation spatiales et spatio-temporelles (théoriques et appliquées) dans des domaines privilégiés tel que l'environnement, l'écologie, l'épidémiologie et la biologie des populations.
L'organisation hierarchique du laboratoire est simple. Il y a un directeur d'unité, son assistante,

les chercheurs et les ingénieurs :

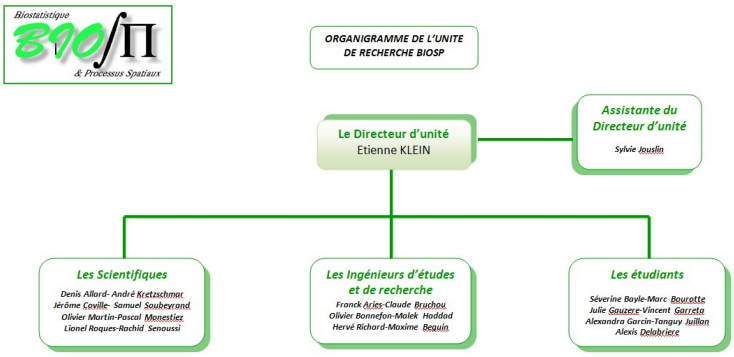


FIGURE 1 – Organigramme du laboratoire BioSP

2.2 WACSgen, un générateur stochastique de données climatiques

Dans un premier temps, le générateur stochastique de données climatiques WACSgen s'inscrivait dans le projet national CLIMATOR. Ce dernier avait pour but de fournir des méthodes et des résultats sur l'impact du changement climatique sur des systèmes cultivés variés, à l'échelle de la parcelle, et dans des climats contrastés français. L'approche, qui consistait en un exercice de modélisation pour de nombreux systèmes de culture (blé, tournesol,maïs, forêt,..), a été faite sur treize sites représentatifs de ces climats : Avignon, Bordeaux, Clermont Ferrand, Colmar, Mirecourt, Mons, Lusignan, Rennes, Sain Etienne, Toulouse, Versailles, et Guadeloupe. Le travail réalisé dans CLIMATOR reposait sur une analyse d'impacts possibles selon divers scénarios pour le climat futur. Il ne s'agit en aucun cas de donner une approche prévisionnelle, mais plutôt de traduire les hypothèses climatiques en impacts chiffrés sur l'agriculture et la forêt françaises.

Dans ce contexte, D. Allard s'occupait des analyses statistiques, des incertitudes et de la création d'un générateur stochastique de temps. Aujourd'hui encore, ce générateur est sans cesse amélioré. WACSgen (Weather-state Approach Conditionaly Skewed generator), écrit en langage R, permet de créer des séries de données climatiques journalières, conditionnellement aux séries observées dans le passé sur les sites voulus. Ainsi les séries simulées doivent posséder les mêmes caractéristiques statistiques que les séries observées.

Il est basé sur l'estimation de différents types de temps pour chaque saison, et utilise les chaînes de Markov pour modéliser le passage entre les types de temps. Pour chaque état de temps, on fait l'hypothèse que les résidus de toutes les variables suivent une loi Complete Skew-Normal (`CSN`) multivariée. Une `CSN` est une variable aléatoire dont la densité est obtenue par la multiplication d'une densité Gaussienne multivariée et d'une fonction de répartition d'une autre Gaussienne multivariée. Pour un vecteur Y de dimension k et qui suit une loi $CSN_{k,l}(\mu, \Sigma, D, \nu, \Delta)$, sa densité sera :

$$f_{k,l}(\mathrm{y}) = c_l \phi_k(\mathrm{y}, \mu, \Sigma)\Phi_l(D(\mathrm{y}-\mu); \nu; \Delta) \text{ avec } c_l^{-1} = \Phi_l(0; \nu, \Delta + D\Sigma D')$$

où :

- $\mu \in \mathbb{R}^k$ et $\nu \in \mathbb{R}^l$ sont deux vecteurs de moyennes,
- $\Sigma \in \mathbb{R}^{k*k}$ and $\Delta \in \mathbb{R}^{l*l}$ sont deux matrices de covariance,
- $\phi_k(\mathrm{y}, \mu, \Sigma)$ et $\Phi_k(\mathrm{y}; \mu; \Sigma)$ sont respectivement la densité et la fonction de répartition d'une loi Normale multivariée avec μ comme vecteur de moyenne et Σ comme matrice de covariance,
- D' est la transposée de la matrice de passage D, de dimension $l*k$. D porte les paramètres de skewness. Lorsque D=0, la CSN devient une loi Gaussienne multivariée $N_k(\mu; \Sigma)$.

Pour simplifier, on décide de poser :

- $k = l$ et $\nu = 0$: avoir un paramètre égal à 0 simplifie toujours les calculs,
- $\Delta = I_k - S^2$ où S est composée des coefficients de skewness (compris entre -1 et 1) pour chaque variable et I_k est la matrice identité de dimension $k * k$: $I_k - S^2$ est une matrice diagonale,
- $D = S\Sigma^{\frac{-1}{2}}$ où $\Sigma^{\frac{-1}{2}}\Sigma^{\frac{-1}{2}} = \Sigma^{-1}$.

Ces choix permettent d'obtenir : $c_l^{-1} = 2^{-k}$

On obtient donc pour la densité :

$$f_{k,k}(\mathrm{y}) = 2^k \phi_k(\mathrm{y}, \mu, \Sigma)\Phi_l(S\Sigma^{\frac{-1}{2}}(\mathrm{y}-\mu); 0; I_k - S^2)$$

et pour la loi : $CSN_k(\mu, \Sigma, S\Sigma^{\frac{-1}{2}}, 0, I_k - S^2)$. Pour simplifier les notations on pose :

$$CSN_k(\mu, \Sigma, S\Sigma^{\frac{1}{2}}, 0, I_k - S^2) = CSN *_k (\mu, \Sigma, S)$$

On modélise les résidus Y_d de nos variables de la façon suivante :

$$\begin{pmatrix} Y_d \\ Y_{d+1} \end{pmatrix} \sim \mathrm{CSN}^* \left(\begin{pmatrix} \mu_d \\ \mu_{d+1} \end{pmatrix}, \begin{pmatrix} \Sigma_d & \Sigma_d R \\ R\Sigma_d & \Sigma_{d+1} \end{pmatrix}, \begin{pmatrix} S_d \\ S_{d+1} \end{pmatrix} \right)$$

où :
Y_d est le vecteur contenant les résidus des variables observées au jour d ,
Y_{d+1} est le vecteur contenant les résidus des variables observées au jour $d+1$,
μ_d est le vecteur de moyennes des résidus observées au jour d,
μ_{d+1} est le vecteur de moyennes des résidus observées au jour $d+1$,
Σ_d est la matrice de covariance des résidus au jour d,
Σ_{d+1} est la matrice de covariance des résidus au jour $d+1$,
R est la matrice diagonale des coefficients de corrélations,
S_d est le vecteur des coefficients de skewness des variables observées au jour d,
et S_{d+1} est le vecteur des coefficients de skewness des variables observées au jour $d+1$.

Pour pouvoir utiliser le générateur, les données doivent être standardisées. La fonction `WACSdata` permet d'uniformiser les données.
L'utilisation de la fonction `WACSdata` se fait de cette manière :

– Les données doivent être rangées en colonne, et doivent suivre cette ordre :
`Année Mois Jour Pluie V1 V2 V3 ... Vn`
La pluie doit toujours être la quatrième colonne et le nombre de variables n'est pas limité ;
– La limite des saisons doit être initialisée. Ainsi, si l'on souhaite baser notre générateur sur les saisons printemps, été, automne et hiver on posera :
`saison.limits = c(60,152,244,365)`
Chaque nombre correspond au premier jour de la ième-saison.
– Si on considère `V1` comme étant la température minimum et `V2` comme étant la température maximum, on peut choisir de remplacer la sixième colonne par la différence entre la température maximum et la température minimum, avec comme condition que la température maximum soit bien supérieure à la température minimum. Dans ce cas, on précise :
`Tminmax = TRUE`

On obtient donc : `data.WACSdata=WACSdata(DATA,season.limits,Tminmax=T)` ou `data.WACSdata=WACSdata(DATA,season.limits,Tminmax=F)` suivant l'option choisi.

Une fois les données standardisées, les paramètres des ces données doivent être estimés. Pour cela, on utilise la fonction `WACSestim`.
`WACSestim` estime :
– la tendance et le coefficient de saisonnalité pour chaque jour et pour chaque variable (excepté la pluie) ;
– les paramètres du modèle choisi (soit `Gamma` soit `AB`) pour la pluie ;
– la matrice de transition des états de temps de chaque saison ;
– la moyenne, le coefficient de corrélation et le coefficient de skewness des résidus, pour chaque variable, chaque saison et pour chaque état de temps ;
– et la matrice des covariances des variables pour chaque saison et pour chaque état de temps.

Le nombre d'état de temps pour chaque saison est défini par le paramètre `Nclusters`. Si `Nclusters=NULL` , `WACSestim` estime également le nombre d'états de temps pour chaque type de temps et pour chaque saison. Le générateur fait une distinction entre les jours secs et les jours humides. Ainsi, si on initialise `Nclusters=3`, on obtiendra six (trois fois deux) états de temps par saison.
La transition entre les états de temps peut se faire de façon "soft" ou "hard" :
– si la transition se fait de façon "soft" : chaque jour a une certaine probabilité d'appartenir ou non à un état de temps ;
– si la transition se fait de façon "hard" : chaque jour appartient à un état de temps unique.
Ceci est défini par le paramètre `clustering`.

On obtient donc :
`data.par=WACSestim(data.WACSdata,spar=0.7,trend.norm="L2",rain.model,method="MOM",vsel=NULL, Nclusters,clustering,plot.it=F,DIR="")`
La méthode d'estimation que j'ai utilisé tout au long de mon stage a été celle de la méthode des moments et le modèle pour la pluie a été le modèle Gamma.

Une fois les paramètres estimés, on peut simuler les données avec la fonction `WACSsimul`. Pour utiliser cette fonction, on doit initialisé :
– le nombre d'années que l'on souhaite simuler ;

– un vecteur contenant les limites des différentes saison, `season.limits` ;
– un vecteur contenant les bornes de chaque variable (excepté la pluie), `vbounds`, par exemple :
`vbounds = cbind(c(-25,0,0,0),c(25,30,3600,20))`, ainsi ici on dit que la température minimum est comprise entre -25 et 25 degrès.

On obtient donc :

```
data.simul=WACSsimul(Ny,season.limits,vbounds,data.par)
```

2.3 Ma mission, un diagnostic du modèle

Ma mission a été de produire un diagnoctic sur le modèle et son implémentation informatique.

Je me suis d'abord occupée de tester les codes qui permettent de simuler des lois `CSN` : je devais rechercher si il n'y avait pas d'erreurs de programmation en faisant tourner les codes sur des données simples ou simulées.
Puis j'ai vérifié que les estimations des paramètres correspondaient bien aux valeurs voulues en mettant en évidence le ou les cas pour lesquels l'algorithme d'estimation rencontrait des difficultés à obtenir de bons résultats.
Et enfin j'ai testé le modèle sur les données relevées quotidiennement sur trois sites : Colmar, Avignon et Toulouse ; en considérant cinq variables : la pluie observée (`Rain`),la température minimum (`Tmin`), la température maximum (`Tmax`), la radiation totale (`Radiation`) et la vitesse moyenne du vent (`Wind`).

Tout au long du stage, j'ai travaillé sous :
– Le langage R, pour créer des fonctions,
– RStudio, pour valider les codes et mettre en place les simulations,
– Word, pour produire les documents de synthèse et stocker les résultats obtenus,
– LaTex, pour rédiger mon rapport.

3 Travail effectué

3.1 Méthode de travail

3.1.1 Validation des routines

J'ai commencé par tester les codes de simulation d'une loi `CSN`. Deux fonctions ont été créées pour simuler une `CSN` : une fonction qui simulait une `CSN` générale et une qui simulait une `CSN*`. Mon premier travail a donc été de trouver à quel moment les deux lois simulées étaient différentes, pour quelle valeur des paramètres. Pour ce faire, j'ai utilisé la méthode suivante :

- paramètres identiques pour les deux lois : `mu=0, skew varie, covariance varie, correlation varie, NombreVariable=2`, tel que, pour deux variables simulées :

  `mu` (répétée le nombre de variables de fois) correspond au vecteur μ :

  $$\mu = (\texttt{mu}, \texttt{mu})$$

  `skew` (répété le nombre de variables de fois) correspond à la diagonale de la matrice S :

  $$S = \begin{pmatrix} \texttt{skew} & 0 \\ 0 & \texttt{skew} \end{pmatrix}$$

  `covariance` (répétée le nombre de variables de fois au carré moins le nombre de variables) correspond aux valeurs en dessous et au dessus de la diagonale de la matrice Σ :

  $$\Sigma = \begin{pmatrix} 1 & \texttt{covariance} \\ \texttt{covariance} & 1 \end{pmatrix}$$

  `correlation` (répétée le nombre de variables de fois) correspond à la diagonale de matrice R :

  $$R = \begin{pmatrix} \texttt{correlation} & 0 \\ 0 & \texttt{correlation} \end{pmatrix}$$

- 1000 simulations des deux lois (afin d'avoir des résultats significatifs) ;
- test de Kolmogorov Smirnov pour tester si les deux lois simulées suivent la même loi (test d'ajustement sur variable continue) ;
- opération répété 50 fois afin d'avoir 50 p values pour chaque valeur prise par le ou les paramètres variant ;
- calcul de la moyenne des p values pour chaque valeur prise par le ou les paramètres variant,
- graphique des moyennes des pvalues obtenues,

J'ai fait de même avec le code permettant de simuler une loi `CSN` conditionnelle, afin de vérifier que la loi suivie par Y_{d+1} était bien une loi `CSN`. J'ai d'abord fait varier un paramètre (skew, covariance ou corrélation) de -0.9 à 0.9, et j'ai mis à égalité les deux autres (0.5, -0.5 ou 0). J'ai continué en faisant varier deux paramètres en sens inverse.

J'ai recommencé les simulations en appliquant la même méthode, et j'ai comparé les pvalues obtenues pour les données simulées normales et les données simulées centrées. J'ai fait cela pour différentes versions du code, jusqu'à ce que l'on obtienne des résultats satisfaisants.

Le test de Kolmogorov Smirnov ne peut pas s'appliquer dans le cas multivarié. Malgré tout j'ai appliqué ce test à chaque colonne distinctement (pour chaque variable). J'ai cherché des tests applicables au cas multivarié mais je n'en ai trouvé aucun applicable sous R.

Les résultats ainsi obtenus permettaient quand même de déceler des éventuelles erreurs de programmation.

3.1.2 Validation du modèle mathématique

La deuxième partie de mon travail consistait à tester les estimations des paramètres faites par la fonction d'estimation, `estim.csnstar`.

Pour cela, j'ai généré N=1000 simulations des deux vecteurs Y_d et Y_{d+1} à 4 variables, correspondant à deux jours successifs, selon le schéma suivant :

On simule une première fois : $(Y_d, Y_{d+1}) = (V_1^d, V_2^d, V_3^d, V_4^d, V_1^{d+1}, V_2^{d+1}, V_3^{d+1}, V_4^{d+1})$
Puis une deuxième : $(Y'_d, Y'_{d+1}) = (V_1'^d, V_2'^d, V_3'^d, V_4'^d, V_1'^{d+1}, V_2'^{d+1}, V_3'^{d+1}, V_4'^{d+1})$
On transforme de façon à obtenir le tableau suivant à 2N lignes :

$$N' = \begin{pmatrix} V_1^d & V_2^d & V_3^d & V_4^d \\ V_1^{d+1} & V_2^{d+1} & V_3^{d+1} & V_4^{d+1} \\ V_1'^d & V_2'^d & V_3'^d & V_4'^d \\ \dots & \dots & \dots & \dots \end{pmatrix}$$

Lorsque l'on fera l'estimation de la corrélation sur la suite des 2N vecteurs à 4 variables, la valeur obtenue correspondra à l'estimation de la moitié de la corrélation.

J'ai estimé cent fois ces paramètres. J'ai stocké chaque estimation des paramètres dans des matrices. Je me suis concentrée sur l'estimation des moyennes,de la corrélation et du coefficient de skewness. Chaque matrice représente l'estimation d'un paramètre pour les quatre variables. Par exemple, les cent estimations des coefficients de corrélations pour la première variable sera la colonne une de ma matrice de stockage.

Comme on souhaitait avoir une valeur unique pour l'estimation de la covariance, j'ai fait la moyenne des valeurs contenues sous la diagonale de la matrice de covariance et j'ai stocké cette valeur dans un vecteur pour chaque nouvelle estimation des paramètres.

J'ai ensuite tracé l'histogramme des valeurs obtenues pour chaque variable et pour chaque paramètre en mettant en évidence la valeur de départ. En sortie R, j'ai donc obtenu quatre histogrammes pour chaque paramètre.

Cette méthode a été utilisée pour certains cas :
- un skew fort et une corrélation forte,
- un skew fort et une corrélation faible,
- un skew faible et une corrélation forte,
- un skew faible et une corrélation faible,
- une covariance assez forte de 0.6,
- une covariance nulle,
- un skew nul.

En fonction des résultats obtenus, une deuxième version du code a été créée, `estim.csnstar.v2`. J'ai ensuite produit un document word servant à comparer les deux façons d'estimer, et nous avons pu conserver la version optimale.

Pour avoir une autre échelle de comparaison, j'ai regardé si les différences entre les paramètres estimés et les paramètres initialisés étaient flagrantes graphiquement. J'ai donc simulé une loi marginale `CSN` à l'aide de la fonction `dmcsn.marg` et des paramètres estimés. J'ai ensuite tracé le graphique de la loi simulée `CSN` du départ et j'ai rajouté les contours de la loi marginale.

3.1.3 Confrontation aux données

Une fois les codes validés, j'ai confronté le modèle WACSgen à des données climatiques mesurées. J'ai comparé les données simulées par la fonction `WACSsimul` avec les données observées sur Colmar, Avignon et Toulouse.

Je me suis aidée des fonctions de comparaison créées par Hugo Le Panse, un ancien étudiant en STID qui a fait son stage à l'INRA, en 2010. J'ai optimisé les paramètres de ses fonctions afin de les rendre plus faciles d'utilisation et j'en ai créé d'autres. Les fonctions définis dans R que j'ai utilisé sont : `plot, lines, points, sample, hist`. Les fonctions créées par Hugo Le Panse que j'ai modifié sont : `boxplot.var, summary.var, temporal.correlation, persistence, probability.occurence, correlation.two.var`.
`sum.temp.X` et `confidence.intervals` sont les fonctions que j'ai créé.
Chaque document de comparaison est présenté de la même façon.

J'ai d'abord commencé par faire une comparaison univariée. Pour chaque variable j'ai réalisé :

- Le graphique des données observées grâce à la fonction `plot`. Comme il y avait beaucoup de données à représenter, j'en ai sélectionné dix pour cent à l'aide de la fonction `sample`. J'ai rajouté la réprésentation des données simulées en conservant les mêmes numéros de lignes que pour les données observées ;
- Les boxplots des moyennes et des écarts-types par mois pour les données observées et simulées avec la fonction `boxplot.var`. Pour mieux situer les différences, j'ai conservé les données contenues dans les boxplots de la moyenne et de l'écart-type des données observées et j'ai ajouté ces valeurs sur les boxplots des données simulées (`lines`) ;
- Le graphique des corrélations temporelles des données observées auquel j'ai rajouté les corrélations temporelles des données simulées. J'ai également tracé la droite `y=x` et la droite de régression linéaire entre les données simulées au jour `d` et les données simulées au jour `d+1`. Pour obtenir les coefficients de régression, j'ai utilisé la fonction `lm` ;
- Les boxplots des corrélations temporelles des données observées et des données simulées par mois grâce à la fonction `temporal.correlation` . Pour mieux situer les différences, j'ai conservé les données contenues dans le boxplot des données observées et j'ai ajouté ces valeurs sur les boxplots des données simulées (`lines`). Dans la console s'affiche un résumé statistique des corrélations temporelles calculées pour les deux types de données ;
- Une sortie R indiquant la moyenne, l'écart type et l'autocorrélation globale pour les deux types de données grâce à la fonction `summary.var` ;
- Un graphique représentant la moyenne, l'écart type et l'autocorrélation moyens par mois pour les données observées et simulées grâce à la fonction `summary.var` ;

Vient ensuite une comparaison bivariée où pour chaque couple de variable j'ai produit :

- Le graphique de la première variable en fonction de la deuxième variable pour les données observées. J'ai ensuite rajouté le nuage de point de la première variable en fonction de la

deuxième variable pour les données simulées ;
- Les boxplots des corrélations entre la première et la deuxième variable des données simulées et observées par mois grâce à la fonction `correlation.two.var`.

A l'aide de la fonction `probability.occurence`, j'ai réalisé plusieurs histogrammes représentant la probabilité d'occurence d'un évènement. La fonction a été utilisée pour calculer la probabilité d'occurence que :
- la température maximum soit égale ou supérieure à 10, 15, 20, 25 et 30 degrès pour les données observées et simulées :
  sur l'ensemble des données ;
  de Mars à Mai et de Juin à Août ;
- la température minimum soit égale ou inférieure à -10, -5, 0 et 5 degrès pour les données observées et simulées :
  sur l'ensemble des données ;
  de Septembre à Novembre et de Décembre à Février.

Avec la fonction `persistence`, j'ai pu produire des histogrammes montrant la persistence d'un évènement :
- persistence de sécheresse :
  sur l'ensemble des données ;
  de Mars à Mai et de Juin à Août ;
- persistence d'échaudage (température maximum supérieure à 20 degrès) :
  sur l'ensemble des données :
  de Mars à Mai et de Juin à Août :
- persistence de gel :
  sur l'ensemble des données ;
  de Décembre à Février.

Les sommes de température en base X ont été calculées avec la fonction `sum.temp.X`. Par exemple, pour obtenir cette somme en un jour, on doit calculer la somme de la température minimum et de la température maximum en ce jour, puis on divise le résultat par deux. On soustrait le nouveau résultat par la base X choisie. Si le chiffre obtenu est positif on le conserve. On réitère la méthode pour le nombre de jours voulus et on en calcule la somme. Cette fonction a été utilisée pour calculer les sommes en base 6 et 10, pour les périodes de Mars à Mai et de Mars à Juin.
Cette fonction renvoie le boxplot de ces sommes pour les données observées et simulées ainsi que les sommes totales dans la console de R. Les étoiles sur les boxplots représentent la moyenne pour les deux types de données. Calculer cette somme permet, par exemple de savoir quelle quantité d'énergie une plante va recevoir. Si on prend le maïs, la température de base est de 6°C. En dessous du seuil de 6°C, le maïs a une croissance nulle. Pour attendre le stade de floraison, la somme en base 6 doit être au minimum égale à 790.

Pour finir, j'ai créé la fonction `confidence.intervals` qui renvoie les enveloppes de confiances pour les paramètres souhaités pour chaque jour de l'année en simulant N fois une série de données de même longueur que la série mesurée. Ainsi, pour chaque variable, j'ai réalisé les enveloppes de confiance de la moyenne, l'écart-type et de la corrélation temporelle.

Par exemple, pour réaliser l'enveloppe de confiance de la moyenne de la température minimum au niveau 0.05, j'ai simulé 19 fois un jeu de données avec les paramètres estimés. Pour chaque jeu de données j'ai calculé la moyenne de la température minimum pour chaque jour (de 1 à 365). J'ai ensuite tracé le graphique des moyennes pour chaque jour des données observées auquel j'ai ajouté les moyennes de chaque jeu de données. J'ai mis en évidence la moyenne minimum et la moyenne maximum pour chaque jour des données simulées. J'ai alors pu calculé le nombre de jours exacts où la température minimum moyenne des données observées n'étaient pas comprises dans l'intervalle de confiance.

J'ai réalisé plusieurs versions de ces documents, en estimant les paramètres avec `Nclusters` variant de 1 à 4 et en estimant les états de temps pour chaque jour de façon "soft" et "hard".

3.2 Résultats

3.2.1 Validation des routines

J'ai commencé par tester les deux façons de simuler une `CSN`.

On a remarqué que, lorsque la covariance était positive et que les valeurs pour la corrélation ou le skewness étaient assez fortes, les p-values étaient inférieures à 0.05 : les variables issues de ces simulations ne suivaient pas la même loi.
Si la corrélation ou le skewness était proche de 0, les deux lois simulées pouvaient être considérées comme identiques.

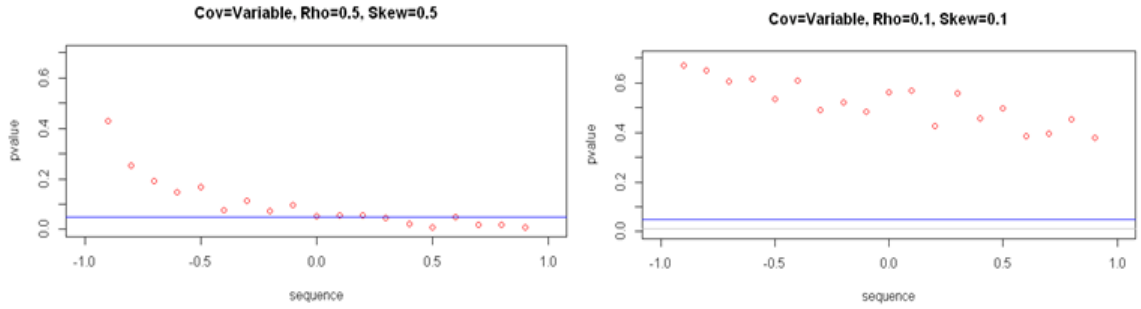


FIGURE 2 – Covariance varie, corrélation et skewness fixes

Grâce aux résultats obtenus, les erreurs de modélisation et de code ont été corrigés. En faisant tourner le programme, toutes les p values étaient supérieures à 0.05. On arrivait donc bien à simuler les mêmes lois pour une variable issue d'une `CSN` et une variable issue d'une `CSN*`.

J'ai utilisé le même programme pour comparer la loi de la variable issus d'une `CSN` avec celle d'une variable issue d'une `CSN` conditionnelle. Je me suis concentrée sur les valeurs des paramètres qui ont posé problème précédemment, à savoir : une corrélation forte, une covariance positive et un skewness fort. J'ai d'abord fait varier le skewness (de -0.9 à 0.9) et la corrélation (de 0.9 à -0.9), puis juste le skewness (de -0.9 à 0.9). J'ai obtenu les graphiques suivants :

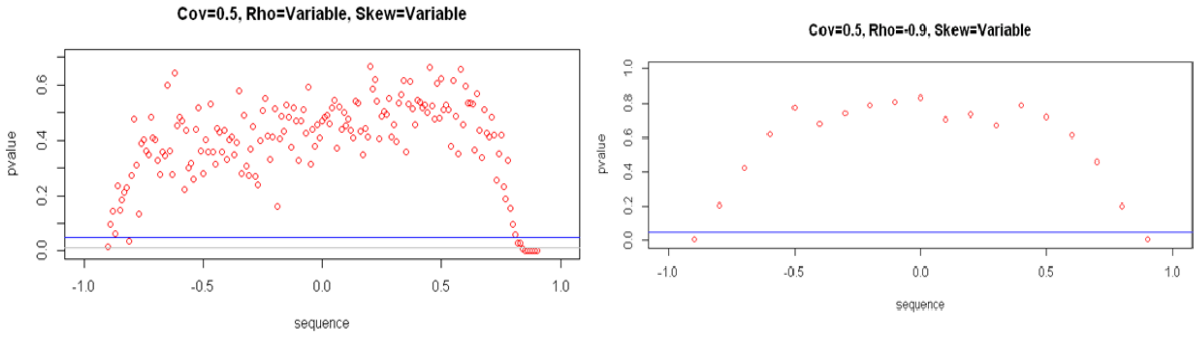


FIGURE 3 – Covariance à 0.5, corrélation variable ou à -0.9 et skewness variable

Lorsque le skewness et la corrélation étaient importants, les deux lois n'étaient pas les mêmes. Pour affiner ce résultat, j'ai tracé l'histogramme des p-values lorsque le skewness était très grand (0.95), et la corrélation et la covariance égales à 0.5. Comme le test de Kolmogorov Smirnov teste les paramètres des lois, comme la moyenne, nous avons décidé de voir si la différence entre les deux lois pouvait être due à la différence des moyennes entre les deux simulations. Voici ce que nous avons obtenu :

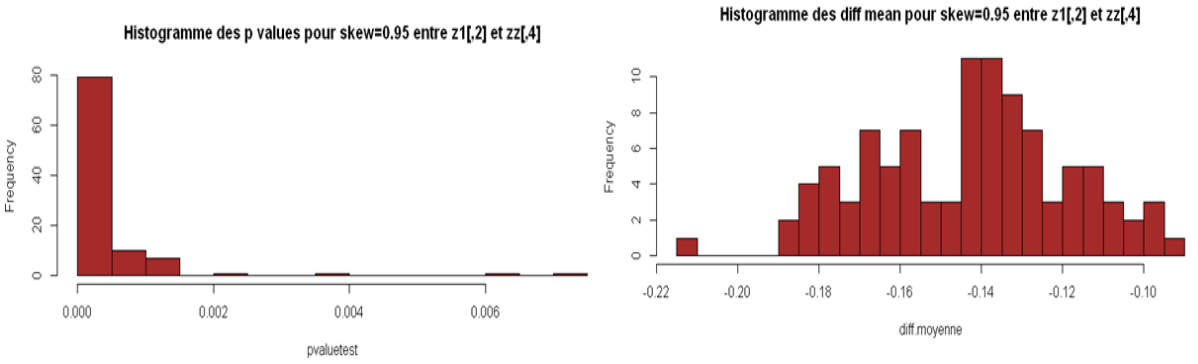


FIGURE 4 – Histogramme des p values et des différences de moyenne

Comme l'écart entre les moyennes était important, on en a déduit que c'était l'une des raisons du rejet de l'hypothèse que les deux lois étaient identiques. Le problème a été situé et les codes ont pu être corrigés.

3.2.2 Validation du modèle mathématique

Une fois les codes de simulation d'une loi `CSN` testés, nous sommes passés au modèle mathématique. J'ai fais tourner la fonction d'estimation des paramètres sur des simulations de loi `CSN` avec des

paramètres que je pouvais controler. Voici deux exemples de résultats obtenus pour la première version du code :

Nom paramètre	Valeur initialisée	Estimation faite par le programme
Mu	0	0.46 ; 0.47 ; 0.48 ; 0.48
Rho	0.7	0.46 ; 0.04 ; 0.06 ; 0.09
Skew	0.8	0.8 ; 0.76 ; 0.8 ; 0.75
Covariance	0.2	0.97 0.17 0.19 0.19 0.17 0.98 0.19 0.15 0.19 0.19 0.92 0.14 0.19 0.15 0.14 0.90
Mu	0	0.05 ; 0.05 ; 0.06 ; 0.05
Rho	-0.9	-0.55 ; -0.43 ; -0.45 ; -0.44
Skew	0.1	0 ; 0 ; 0 ; 0
Covariance	0.8	1.02 0.83 0.81 0.81 0.83 1.01 0.80 0.81 0.81 0.80 0.96 0.79 0.81 0.81 0.79 0.99

FIGURE 5 – Premières estimations

On pouvait voir que :
- la moyenne n'était pas bien estimée,
- la corrélation estimée était assez proche de la corrélation initialisée lorsque la covariance était assez grande,
- la covariance était bien estimée. Cependant quelque fois, la diagonale de la matrice de covariance était supérieure à 1, ce qui normalement n'est pas possible,
- le skewness était assez bien estimé.

Pour avoir un ordre d'idée des valeurs obtenues pour l'estimation, j'ai mis en place une boucle me permettant d'obtenir plusieurs estimations. Ainsi, j'ai pu tracé un histogramme des valeurs estimées pour chaque paramètre et chaque variable. En rouge j'ai représenté la moyenne des estimations et en bleu la valeur initialisée pour simuler la loi `CSN`.
Voici un exemple de résultat obtenu pour l'estimation du skewness (valeur initiale : 0.9) :

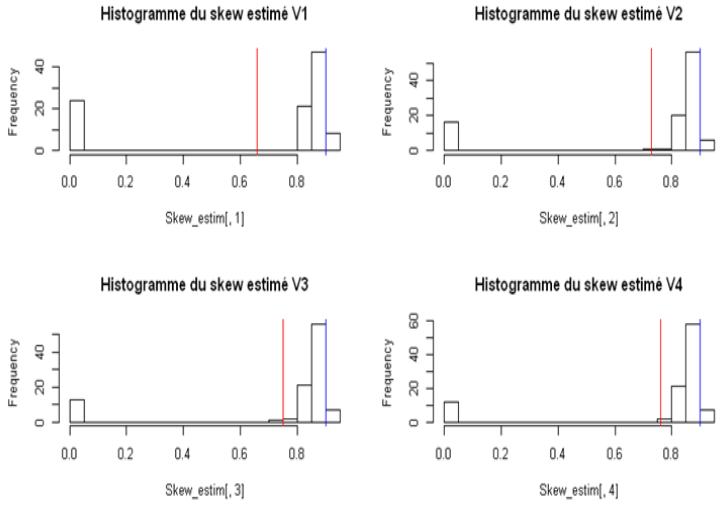


FIGURE 6 – 100 estimations du skewness

Quelque fois, la fonction d'estimation n'était pas stable : pour un skewness très grand, elle pouvait estimer un skewness à 0.
Mais en règle générale, la covariance et la corrélation était bien estimées. Seule la moyenne posait problème. En modifiant un peu le code, on est arrivé à réduire les différences entre la valeur estimée et la valeur réelle. Comme l'ensemble des estimations étaient satisfaisantes, je pouvais commencer les simulations sur les données récoltées.

3.2.3 Confrontation aux données

J'ai commencé par simuler avec les données récoltées sur le site de Colmar. J'ai produit plusieurs documents, en prenant pour paramètre pour la fonction `WACSestim` : `clustering` = "soft" ou "hard", et en faisant varier le `Nclusters` de 1 à 4 ou en le mettant comme NULL. J'ai choisi de prendre `Nclusters` = 2 car je trouvais que les résultats étaient plus satisfaisants. Pour le `clustering` il n'y avait pas de différences flagrantes entre "soft" et "hard".

J'ai donc simulé les données d'Avignon et de Toulouse en prenant : `Nclusters` = 2 et `clustering` = "soft" ou "hard". En comparant les données observés pour chaque site et les données simulées, j'ai obtenu les résultats suivants :

- le nuage de point des données simulées se rapproche de celui des données observées pour toutes les variables :

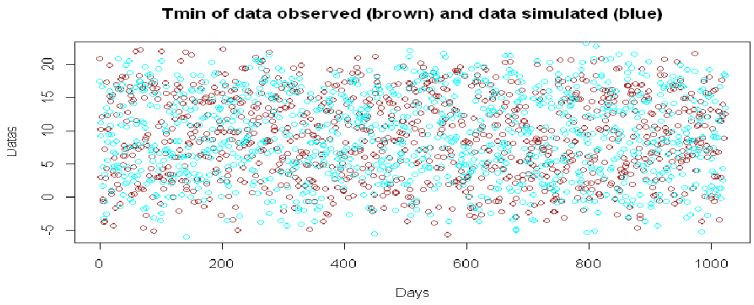


FIGURE 7 – Site Avignon - Variable Tmin - Nclusters = 2 - clustering = soft

– les boxplots des moyennes et des écart types par mois ont montré que les moyennes étaient assez bien simulées, mais les écart types des données simulées sont légèrement supérieurs à ceux des données observées :

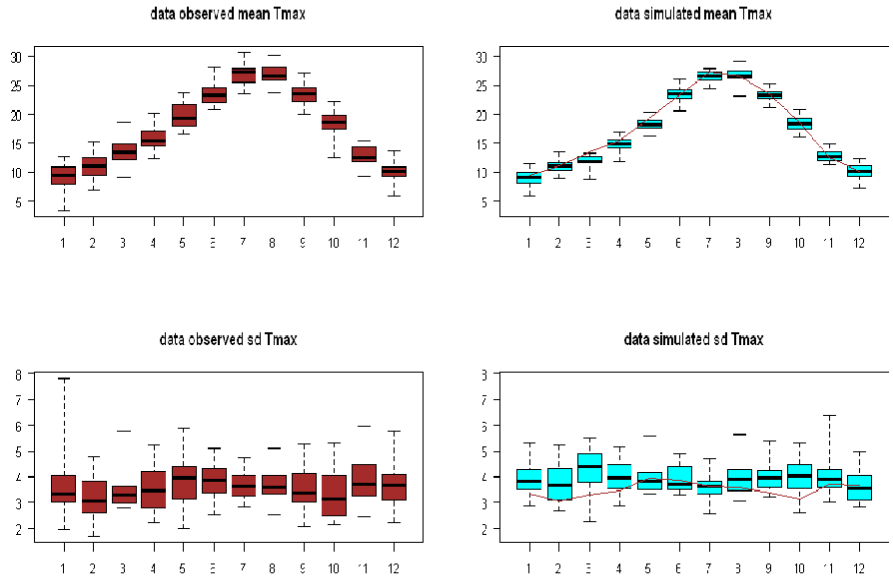


FIGURE 8 – Site Toulouse - Variable Tmax - Nclusters = 2 - clustering = hard

– les corrélations temporelles des données simulées sont encore à améliorer :

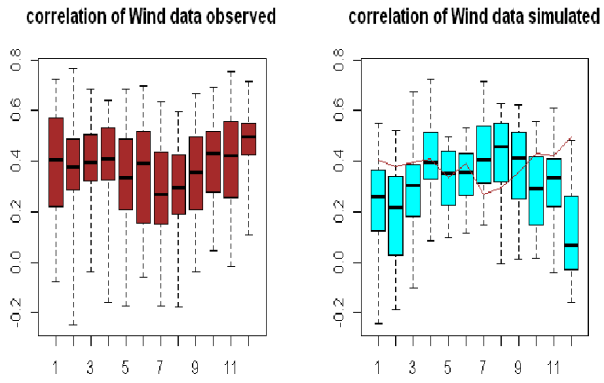


FIGURE 9 – Site Colmar - Variable Wind - Nclusters = 2 - clustering = soft

– la corrélation entre les variables `Tmin/Tmax`,`Tmin/Wind`,`Tmax/Radiation` et `Radiation/Rain` est bien respectée :

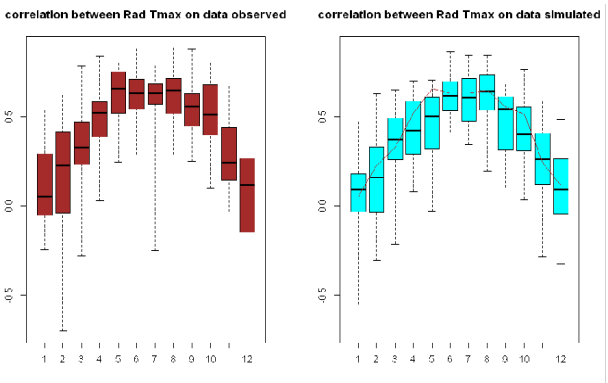


FIGURE 10 – Site Colmar - Variables Radiation/Tmax - Nclusters = 2 - clustering = hard

– les probabilités d'occurence des évènements sont satisfaisantes, la différence entre les données simulées et les données observées se situe au centième voir au millième près :

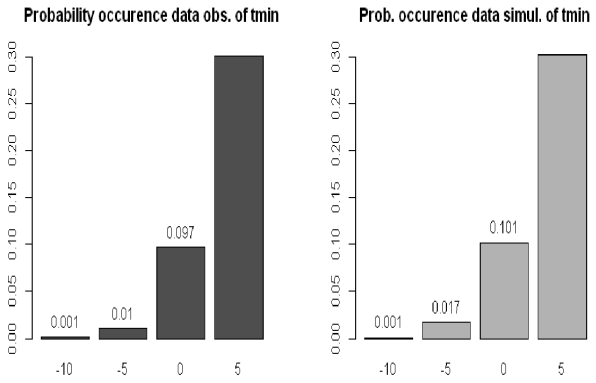


FIGURE 11 – Site Toulouse - Tmin -10,-5,0,5 - Nclusters = 2 - clustering = soft

– pour la persistence d'un évènement, le générateur a du mal à simuler des grandes valeurs. Il simule, par exemple, de nombreux jours de gel, mais ces jours ne sont pas ou peu consécutifs. Mais le nombre de jours total de gel dans les données simulées reste assez proche de celui des données observées :

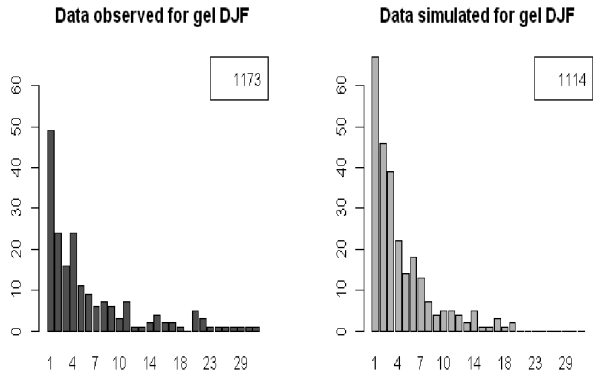


FIGURE 12 – Site Colmar - Gel de Décembre à Février - Nclusters = 2 - clustering = soft

– le boxplot des sommes de température montre que les sommes par an sont plus dispersées dans les données observées que dans les données simulées, mais les moyennes sont assez proches. Dans cet exemple la moyenne pour les données observées est de 307.22 par an et pour les données simulées 270.91 :

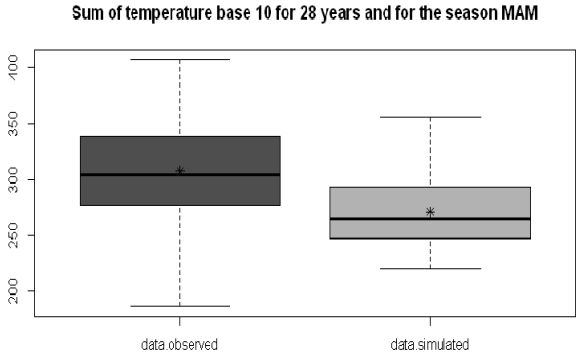


FIGURE 13 – Site Avignon - Base 10 de Mars à Mai - Nclusters = 2 - clustering = hard

– les intervalles de confiance pour les paramètres de chaque variable sont satisfaisants. Il n'y a que la corrélation temporelle que le générateur a du mal à reproduire. En grande majorité, la moyenne des données observées pour chaque paramètre et pour chaque jour de l'année sort de l'enveloppe de confiance entre 0 et 100 jours :

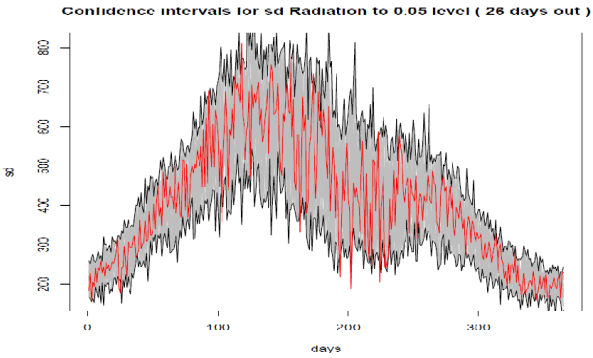


FIGURE 14 – Site Avignon - Enveloppes de confiance écart type Radiation alpha = 0.05 - Nclusters = 2 - clustering = soft

On peut donc conclure que le générateur permet d'avoir des données possédant approximativement la même moyenne et le même écart type que les données observées, mais la corrélation temporelle n'est pas entièrement respectée.

3.3 Discussion du modèle

La fonction `WACSestim` n'arrive pas toujours à estimer les paramètres lorsque l'on demande 3 ou 4 états de temps. Ceci est du fait que l'on ne possède pas assez de données dans chaque état de temps. Le calcul de paramètre est donc plus compliqué. Ce problème est d'autant plus récurrent si l'on dit que un jour correspond à un seul état de temps, c'est à dire si `clustering`="hard".

Sur les graphiques des enveloppes de confiance, on voit les changements de saison. La transition ne devrait pas être aussi claire. Ci dessous deux exemples avec, entourés en bleu, les

moments où la transition n'est pas assez lissée :

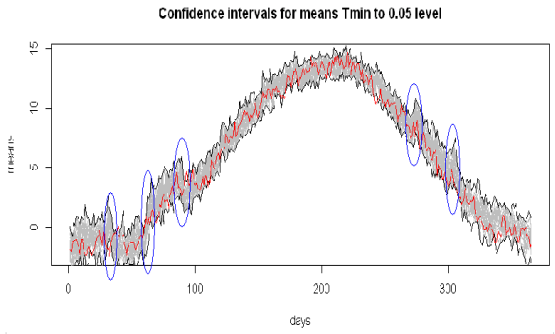


FIGURE 15 – Site Colmar - Enveloppes de confiance moyenne Tmin alpha = 0.05 - Nclusters = 2 - clustering = soft - une saison = un mois

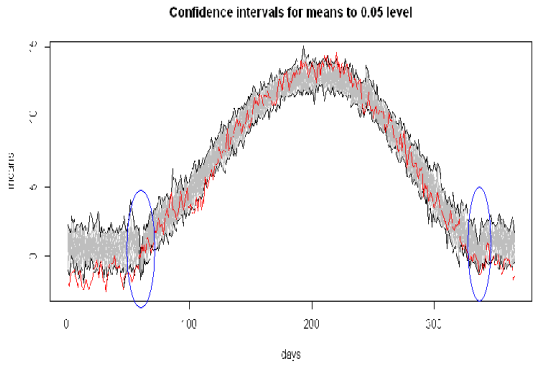


FIGURE 16 – Site Colmar - Enveloppes de confiance moyenne Tmin alpha = 0.05 - Nclusters = 2 - clustering = soft - saisons = nos saisons

Il reste donc du travail à faire de ce point de vu la car on pourrait s'attendre à avoir de meilleurs résultats en augmentant le nombre de saisons.

Pour la pluie, le modèle choisi n'est peut être pas le plus optimal. Je n'ai pas essayé de simuler avec le modèle "AB", modèle proposé par M. Allard et M. Bourotte. Ce modèle est récent et n'a pas encore été implanté dans le générateur WACSgen. Peut être qu'il faudrait étudier cette piste.

4 Conclusion

Ce stage m'a permis d'approfondir mes connaissances en mathématiques. J'ai appris une nouvelle loi (`CSN`) et le principe d'une chaîne de Markov.
Je me suis également améliorée dans la compréhension écrite en anglais, grâce à la lecture d'articles scientifiques (A DAILY WEATHER GENERATOR FOR USE IN CLIMATE CHANGE STUDIES, de C.G. Kilsby et MULTISITE DOWNSCALING OF DAILY PRECIPITATION WITH A STOCHASTIC WEATHER GENERATOR, de D.S Wilks). J'ai ainsi appris de nouveaux mots de vocabulaire scientifique.

En programmation, je sais maintenant créer des fonctions sous R. J'ai appris également à écrire en LaTeX, ce qui a été très utile pour la rédaction de mon rapport. En lisant ces articles, j'ai compris que savoir rédiger en LaTeX était un plus car c'est le langage universel d'écriture dans la communauté des mathématiciens et des informaticiens.

J'ai également découvert le monde de la recherche. J'ai pu comprendre le fonctionnement d'un laboratoire de recherche, ainsi que les intéractions entre tous les laboratoires et tous les chercheurs. Ces derniers doivent être constamment en liaison pour s'entraider.
J'ai pu assisté à des réunions et à des conférences. Ces dernières m'ont beaucoup plu. Même si je ne comprenais pas toujours tout ce que les orateurs nous expliquaient, j'ai pris plaisir à découvrir leur travail.

En créant des documents et des scripts R, j'ai compris qu'une méthodologie s'imposait. Tout doit être en permanence bien libellé, et bien rangé dans chaque dossier. J'ai compris également qu'il est nécessaire d'avoir une présentation claire et précise des fichiers et des scripts R. La rédaction de rapports reprenant la démarche et les résultats est primordiale. Il faut toujours se mettre à la place de la personne qui va nous relire et qui va être obligée de nous comprendre.

Mais le plus important pour moi est que le stage m'a permis de me rassurer dans mes choix pour ma carrière future. Je sais que c'est dans les mathématiques que je veux aller. Décrire des évènements, prévoir, et programmer sont les trois aspects que j'aimerai retrouver dans mon métier. J'ai pris énormément de plaisir à venir travailler à l'INRA, et j'espère qu'il y aura une prochaine fois.

5 Annexes

5.1 Validation des routines - codes R utilisés

5.1.1 Différences entre deux façons de simuler une CSN

```
sequence=seq(-0.9,0.9, by=0.1) #justification du pas : les trois paramètres sont
Nomb.pvalue=50               #compris entre -1 et 1 et l'ordinateur n'est pas assez
pvalue=NULL                   #puissant pour calculer plus de valeurs (trop long)
mean.pvaluetest=NULL
  for (i in sequence)
    {
      Nsimul = 1000
      Nvar   = 2
      covar  = 0.1
      rho    = -i+0.05
      skew   = i
      Mu      = rep(0,Nvar)
      Sigma   = matrix(covar,Nvar,Nvar); diag(Sigma) = rep(1,Nvar)
      Rho     = rep(rho,Nvar)
      Skew    = rep(skew,Nvar)
      pvaluetest = NULL
        for (j in 1:Nomb.pvalue)
          {
            z  = rmcsnstar(Nsimul,Mu,Sigma,Skew)
            zz = rwacs(Nsimul,Sigma,Rho,Skew)
            pvaluetest=c(pvaluetest,ks.test(z, zz[,1:2], alternative="two.sided", exact=TRUE)$p.value)
          }
      mean.pvaluetest=mean(pvaluetest)
      pvalue=c(pvalue,mean.pvaluetest)
      print(i)
      }

par(mfrow=c(1,1))
plot(sequence,pvalue, col="red", main="Cov=0.1, Rho=Variable, Skew=Variable", xlim=c(-1,1), ylim=c(0,0.7))
abline(h=0.05, col="blue")
abline(h=0.01,col="grey")
```

5.1.2 Différence entre loi générale et loi conditionnelle d'une CSN

```
sequence=seq(-0.9,0.9,0.01)
Nombre.pvalue=100
mean.pvaluetest=NULL
pvalue=NULL

for (l in sequence) {
    Nsimul = 1000
    Nvar   = 2
    covar  = 0.5
    rho    = l
    skew   = l
    Mu      = rep(0,Nvar)
    Sigma   = matrix(covar,Nvar,Nvar); diag(Sigma) = rep(1,Nvar)
    Rho     = rep(rho,Nvar)
    Skew    = rep(skew,Nvar)
    pvaluetest=NULL
    for (j in 1:Nombre.pvalue) {
        zz      = rwacs(Nsimul,Sigma,Rho,Skew)
        Nv = dim(Sigma)[1]
        if ( (dim(Sigma)[2] != Nv) | (length(Rho) != Nv) | (length(Skew) !=Nv) )
          stop ("rwacs: dimension error")
        SS = matrix(0,2*Nv,2*Nv)
        SS[1:Nv,1:Nv]                   = Sigma
        SS[(Nv+1):(2*Nv),(Nv+1):(2*Nv)] = Sigma
        SS[1:Nv,(Nv+1):(2*Nv)]          = Sigma%*%diag(Rho)
        SS[(Nv+1):(2*Nv),1:Nv]          = diag(Rho)%*%Sigma
        if(any(eigen(SS)$values < 0)) stop ("rwacs: non positive definite model")
        Mu2     = rep(Mu,2)
        Skew2   = rep(Skew,2)
        D2      = diag(Skew2)%*%solve(sqrtm(SS))
        Nu2     = rep(0,2*Nv)
        Delta2  = diag(1,2*Nvar) - diag(Skew2)^2
        z1      = matrix(0,Nsimul,Nvar)
        for (i in 1:Nsimul) {
           z1[i,]  = rmcsn.cond(zz[i,1:Nvar],Mu2,SS,D2,Nu2,Delta2)
           }
        pvaluetest=c(pvaluetest, ks.test(z1[,1], zz[,3], alternative="two.sided", exact=TRUE)$p.value)
    }
    mean.pvaluetest=mean(pvaluetest)
    pvalue=c(pvalue,mean.pvaluetest)
    print(l)
}


par(mfrow=c(1,1))
plot(sequence,pvalue, col="red", main="Cov=0.5, Rho=-0.9, Skew=Variable", xlim=c(-1,-0.7), ylim=c(0,0.5))
abline(h=0.05, col="blue")
abline(h=0.01,col="grey")
```

5.1.3 Simulations loi CSN et différence de moyennes

```
Nombre.pvalue = 100
p.value.c1=NULL
p.value.c2=NULL
p.value.c.centre1=NULL
p.value.c.centre2=NULL
diff.moyenne1=NULL
diff.moyenne2=NULL
for (j in 1:Nombre.pvalue) {
  # valeurs que l'on peut changer à sa guise
  Nsimul = 1000
  Nvar   = 2
  covar  = 0.5
  # Initialisation, selon les valeurs ci-dessus
  Mu      = rep(0,Nvar)
  Sigma   = matrix(covar,Nvar,Nvar); diag(Sigma) = rep(1,Nvar)
  Rho     = c(0.4,0.9)
  Skew    = c(0.5,0.9)
  SS = matrix(0,2*Nvar,2*Nvar)
  SS[1:Nvar,1:Nvar]                       = Sigma
  SS[(Nvar+1):(2*Nvar),(Nvar+1):(2*Nvar)] = Sigma
  SS[1:Nvar,(Nvar+1):(2*Nvar)]            = Sigma%*%diag(Rho)
  SS[(Nvar+1):(2*Nvar),1:Nvar]            = diag(Rho)%*%Sigma
  if(any(eigen(SS)$values < 0)) stop ("rwacs: non positive definite model")
  SS.invsqrtm  = sqrtm(solve(SS))
  Mu2     = rep(Mu,2)
  Skew2   = rep(Skew,2)
  D2      = diag(Skew2)%*%SS.invsqrtm
  Nu2     = rep(0,2*Nvar)
  Delta2  = diag(1,2*Nvar) - diag(Skew2)^2
  zc      = matrix(0,Nsimul,Nvar); zn = zc
  B           = SS.invsqrtm[1:Nvar,1:Nvar]
  C           = SS.invsqrtm[1:Nvar,(Nvar+1):(2*Nvar)]
  Sigma.cond  = Sigma - diag(Rho)%*%Sigma%*%diag(Rho)
  D.cond      = rbind(diag(Skew)%*%C,diag(Skew)%*%B)
  zz      = rwacs(Nsimul,Sigma,Rho,Skew)
  for (i in 1:Nsimul){
    Mu.cond = Mu + diag(Rho)%*%(zz[i,1:Nvar])
    Nu.cond = rbind(diag(Skew)%*%(B + C%*%diag(Rho)),diag(Skew)%*%(C + B%*%diag(Rho)))%*%(Mu-zz[i,1:Nvar])
    zc[i,]  = rmcsn.cond(zz[i,1:Nvar],Mu2,SS,D2,Nu2,Delta2)
  }
  zz.centre1=zz[,3] - mean(zz[,3])
  zc.centre1=zc[,1] - mean(zc[,1])
  zz.centre2=zz[,4] - mean(zz[,4])
  zc.centre2=zc[,2] - mean(zc[,2])
  p.value.c1=c(p.value.c1, ks.test(zc[,1], zz[,3], alternative="two.sided", exact=TRUE)$p.value)
  p.value.c2=c(p.value.c2, ks.test(zc[,2], zz[,4], alternative="two.sided", exact=TRUE)$p.value)
  p.value.c.centre1=c(p.value.c.centre1, ks.test(zc.centre1, zz.centre1, alternative="two.sided", exact=TRUE)$p.value)
  p.value.c.centre2=c(p.value.c.centre2, ks.test(zc.centre2, zz.centre2, alternative="two.sided", exact=TRUE)$p.value)
  diff.moyenne1=c(diff.moyenne1, mean(zz[,3]) - mean(zc[,1]))
  diff.moyenne2=c(diff.moyenne2, mean(zz[,4]) - mean(zc[,2]))
  print(j)
}
par(mfrow=c(1,2))
hist(p.value.c1,main="p values zc[,1] et zz[,3]", nclass=20,col="brown")
hist(p.value.c2,main="p values zc[,2] et zz[,4]", nclass=20,col="brown")
hist(p.value.c.centre1,main="p values zc.centre[,1] et zz.centre[,3]", nclass=20,col="brown")
hist(p.value.c.centre2,main="p values zc.centre[,2] et zz.centre[,4]", nclass=20,col="brown")
hist(diff.moyenne1, main="difference de moyennes entre zz[,3] et zc[,1]", nclass=20, col="brown")
hist(diff.moyenne2, main="difference de moyennes entre zz[,4] et zc[,2]", nclass=20, col="brown")
```

5.2 Validation du modèle mathématique

```
Nsimul = 1000
Nvar   = 4
Mu     = rep(0,Nvar)
covar  = 0
Rho    = rep(0.8,4)
Skew   = rep(0.4,4)
Sigma   = matrix(covar,Nvar,Nvar); diag(Sigma) = rep(1,Nvar)
SS = matrix(0,2*Nvar,2*Nvar)
SS[1:Nvar,1:Nvar]                       = Sigma
SS[(Nvar+1):(2*Nvar),(Nvar+1):(2*Nvar)] = Sigma
SS[1:Nvar,(Nvar+1):(2*Nvar)]            = Sigma%*%diag(Rho)
SS[(Nvar+1):(2*Nvar),1:Nvar]            = diag(Rho)%*%Sigma
if(any(eigen(SS)$values < 0)) stop ("Attention ! modèle non admissible: changer Rho ou covar")
SS.invsqrtm  = sqrtm(solve(SS))
Mu2     = rep(Mu,2)
Skew2   = rep(Skew,2)
Nu2     = rep(0,2*Nvar)
Delta2  = diag(1,2*Nvar) - diag(Skew2)^2
D       = diag(Skew2)%*%SS.invsqrtm
Nombre=100
covar_estim=NULL
Rho_estim=matrix(0, Nombre, Nvar)
Skew_estim=matrix(0,Nombre,Nvar)
Mu_estim=matrix(0,Nombre,Nvar)
for (l in 1:Nombre ) {
    # On génère les données
    YY      = rwacs(Nsimul,Sigma,Rho,Skew)
    Y       = matrix(0,2*Nsimul,Nvar)
    sequence = 1:Nsimul
    for (i in sequence){
      Y[(2*(i-1)+1),] = YY[i,1:Nvar]
      Y[(2*i),] = YY[i,(Nvar+1):(2*Nvar)]
    }
    z      = rep(1,dim(Y)[1])
    day    = 1:dim(Y)[1]
    # On réalise l'estimation
    plot.item=c("essai","23avril")
    toto = estim.csnstar(Y,z,day,plot.it=FALSE,DIR="./",plot.item)
    # On extrait les paramétres estimés
    Skew_hat = toto$skew
    Rho_hat  = 2*toto$rho
    Sigma_hat = toto$cov
    SS_hat = matrix(0,2*Nvar,2*Nvar)
    SS_hat[1:Nvar,1:Nvar]                       = toto$cov
    SS_hat[(Nvar+1):(2*Nvar),(Nvar+1):(2*Nvar)] = toto$cov
    SS_hat[1:Nvar,(Nvar+1):(2*Nvar)]            = toto$cov%*%diag(Rho_hat)
    SS_hat[(Nvar+1):(2*Nvar),1:Nvar]            = diag(Rho_hat)%*%toto$cov
    M  = apply(YY,2,"mean")
    Mu2 =as.vector(M - sqrtm(SS_hat)%*%diag(rep(Skew_hat,2))%*%rep(sqrt(2/pi),2*Nvar))
    Mu_hat = 0.5*(Mu2[1:Nvar]+Mu2[(Nvar+1):(2*Nvar)])
    D_hat = diag(rep(Skew_hat,2))%*%sqrtm(solve(SS_hat))
    Delta_hat = diag(2*Nvar) - diag(rep(Skew_hat,2)^2)
    covar_hat = mean(Sigma_hat[lower.tri(Sigma_hat)])
    covar_estim=c(covar_estim, covar_hat)
    for (j in 1:Nvar) {
      Rho_estim[l,j]=Rho_hat[j]
      Skew_estim[l,j]=Skew_hat[j]
```

```
      Mu_estim[l,j]=Mu_hat[j]
    }
    print(l)
}
```

5.3 Confrontation aux données

5.3.1 Quelques fonctions utilisées

BOXPLOT.VAR

```
boxplot.var = function (var1,var2,name.var) {

  ########################################################
  #
  # WACSgen project v2013. Author C. Flecher, D. Allard
  #
  #   Function boxplot var : main function to create boxplots of data observed
  #    and data simulated (comparison by month)
  #
  #   ARGUMENTS :
  #    var1 :     variable observed
  #    var2 :     variable simulated
  #    name.var : variable name (use for graph)
  #
  #

  if ( length(var1) != length(var2) ) {
    stop ("Please length(var1) and length(var2) must be the same")
  }else{}
  ######################################################################
  Ny           =  length(var1)%/%365
  month        =  rep(c(rep(1,31),rep(2,28),rep(3,31),rep(4,30),rep(5,31),rep(6,30),rep(7,31),
                        rep(8,31),rep(9,30),rep(10,31),rep(11,30),rep(12,31)),Ny)
  year         =   NULL
  for ( i in 1:Ny ) {
    year  =  c(year, rep(i,365))
  }
  var1         =   cbind(year,month,var1)
  var1         =   as.data.frame(var1)
  colnames(var1)  =  c("year","month","var")
  var2         =   cbind(year,month,var2)
  var2         =   as.data.frame(var2)
  colnames(var2)  =  c("year","month","var")
  ######################################################################
  mean.var1  =  matrix(0,Ny,length(unique(var1$month)))
  for ( i in unique(var1$year) ) {
    for ( j in unique(var1$month) ) {
      mean.var1[i,j] =  mean(var1$var[ var1$year == i & var1$month == j ])
    }
  }
  mean.var2  =  matrix(0,Ny,length(unique(var2$month)))
  for ( i in unique(var2$year) ) {
    for ( j in unique(var2$month) ) {
      mean.var2[i,j] =  mean(var2$var[ var2$year == i & var2$month == j ])
    }
```

```
  }
  sd.var1    =  matrix(0,Ny,length(unique(var1$month)))
  for ( i in unique(var1$year) ) {
    for ( j in unique(var1$month) ) {
      sd.var1[i,j]   =  sd(var1$var[ var1$year == i & var1$month == j ])
    }
  }
  sd.var2    =  matrix(0,Ny,length(unique(var2$month)))
  for ( i in unique(var2$year) ) {
    for ( j in unique(var2$month) ) {
      sd.var2[i,j]   =  sd(var2$var[ var2$year == i & var2$month == j ])
    }
  }
  ######################################################################
  par(mfrow=c(2,2))
  boxplot(mean.var1,col="brown",las=1,range=0,main=paste("data observed mean",name.var),
        ylim=c(min(mean.var1,mean.var2),max(mean.var1,mean.var2)))
  g = boxplot(mean.var1,plot=F)
  boxplot(mean.var2,col="cyan",las=1,range=0,main=paste("data simulated mean",name.var),
        ylim=c(min(mean.var1,mean.var2),max(mean.var1,mean.var2)))
  lines(unique(var1$month),g$stat[3,],col="brown",type="l")
  boxplot(sd.var1,col="brown",las=1,range=0,main=paste("data observed sd",name.var),
        ylim=c(min(sd.var1,sd.var2),max(sd.var1,sd.var2)))
  g = boxplot(sd.var1,plot=F)
  boxplot(sd.var2,col="cyan",las=1,range=0,main=paste("data simulated sd",name.var),
          ylim=c(min(sd.var1,sd.var2),max(sd.var1,sd.var2)))
  lines(unique(var1$month),g$stat[3,],col="brown",type="l")
}
```

TEMPORAL.CORRELATION

```
temporal.correlation = function(var1,var2,name.var) {

  ######################################################
  #
  # WACSgen project v2013. Author C. Flecher, D. Allard
  #
  #   Function temporal correlation : main function to create correlation boxplots
  #   between day d et day d+1
  #
  #   ARGUMENTS :
  #    Ny :       number of years was simulated by WACSsimul, start at the first year at the first
  #               season
  #    var1 :     variable observed
  #    var2 :     variable simulated
  #    name.var : variable name (use for graph)
  #

  if ( length(var1) != length(var2) ) {
    stop ("Please length(var1) and length(var2) must be the same")
  }else{}
  ######################################################################
  Ny            =  length(var1)%/%365
  month         =  rep(c(rep(1,31),rep(2,28),rep(3,31),rep(4,30),rep(5,31),rep(6,30),rep(7,31),
                         rep(8,31),rep(9,30),rep(10,31),rep(11,30),rep(12,31)),Ny)
  year          =  NULL
  for (i in 1:Ny) {
    year = c(year, rep(i,365))
  }
  var1.bis      =  var1[-length(var1)]
  var2.bis      =  var2[-length(var2)]
  var1          =  var1[-1]
  var2          =  var2[-1]
  var1          =  cbind(month[-length(month)],year[-length(year)],var1)
  var1          =  as.data.frame(var1)
  colnames(var1)     = c("month","year","var")
  var1.bis      =  cbind(month[-length(month)],year[-length(year)],var1.bis)
  var1.bis      =  as.data.frame(var1.bis)
  colnames(var1.bis) = c("month","year","var")
  var2          =  cbind(month[-length(month)],year[-length(year)],var2)
  var2          =  as.data.frame(var2)
  colnames(var2)     = c("month","year","var")
  var2.bis      =  cbind(month[-length(month)],year[-length(year)],var2.bis)
  var2.bis      =  as.data.frame(var2.bis)
  colnames(var2.bis) = c("month","year","var")
  ######################################################################
  cor.var1  =  matrix(0,Ny,length(unique(var1$month)))
  for ( i in unique(var1$year) ) {
    for ( j in unique(var1$month) ) {
      cor.var1[i,j] = cor(var1$var[ var1$year == i & var1$month == j ], var1.bis$var[ var1.bis$year == i & var1.bis$month == j ]
    }
  }
  cor.var2  =  matrix(0,Ny,length(unique(var2$month)))
  for ( i in unique(var2$year) ) {
    for ( j in unique(var2$month) ) {
      cor.var2[i,j] = cor(var2$var[ var2$year == i & var2$month == j ], var2.bis$var[ var2.bis$year == i & var2.bis$month == j ]
    }
  }
  ######################################################################
```

```
  par(mfrow=c(1,2))
  boxplot(cor.var1,main=paste("correlation of",name.var,"data observed"),ylim=c(min(cor.var1[!is.na(cor.var1)],cor.var2[!is.na(c
          max(cor.var1[!is.na(cor.var1)],cor.var2[!is.na(cor.var2)])),range=0,col="brown")
  h = boxplot(cor.var1,plot=F)
  boxplot(cor.var2,main=paste("correlation of",name.var,"data simulated"),ylim=c(min(cor.var1[!is.na(cor.var1)],cor.var2[!is.na(
          max(cor.var1[!is.na(cor.var1)],cor.var2[!is.na(cor.var2)])),range=0,col="cyan")
  lines(unique(var1$month),h$stat[3,],col="brown",type="l")
  print ( paste("summary of data observed correlation",name.var,":") )
  print ( summary(apply(cor.var1,2,median)) )
  print ( paste("summary of data simulated correlation",name.var,":") )
  print ( summary(apply(cor.var2,2,median)) )
}
```

CONFIDENCE.INTERVALS

```
confidence.intervals = function(data.obs,data.par,name.var,stat,alpha,Tminmax=F) {

  ####################################################
  #
  # WACSgen project v2013. Author C. Flecher, D. Allard
  #
  #   Function confidence intervals : create confidence intervals
  #
  #   ARGUMENTS :
  #    data.obs :     datas observed (file complete)
  #    data.par :     parameters estimate by WACSestim on data.obs
  #    name.var :     variable used for create confidence intervals
  #                   (Rain,Tmin,Tmax,Range,Radiation or Wind)
  #    stat :         statistic will be choosed to compare (mean, sd or correlation)
  #    alpha :        level to decide the number of simulation (level of confidence)
  #    Tminmax :      used to know if in WACSestim user uses Range or Tmax
  #

  if ( dim(data.obs)[2] != 9 ) {
    stop ("Are you sure for data obs ?")
  }
  if ( name.var != "Rain" & name.var != "Tmin" & name.var != "Tmax" & name.var != "Radiation" & name.var != "Wind") {
    stop ("Can you choose Rain, Tmin, Tmax, Radiation or Wind for name.var please")
  }
  if ( stat != "mean" & stat != "sd" & stat != "correlation" ) {
    stop ("Can you choose mean, sd or correlation for stat please")
  }
  if ( alpha != 0.05 & alpha != 0.01 & alpha != 0.1 ) {
    stop ("Can you choose 0.05, 0.01, or 0.1 for alpha please")
  }
  ####################################################################
  Ny          =  dim(data.obs)[1] %/% 365
  day         =  rep(1:365,Ny)
  year        =  NULL
  for ( i in 1:Ny ) {
    year  =  c(year, rep(i,365))
  }
  N  =  ((1 - alpha)*100) / (alpha*100)
  vbounds = cbind(c(-25,0,0,0),c(25,30,3600,20))
  ####################################################################
  data.simul = matrix(0,365*Ny,5*N)
  for ( i in 1:N ) {
    simul = WACSsimul(28,season.limits,vbounds,data.par)
    simul = simul[,4:8]
    data.simul[,(1+(5*(i-1))):(5*i)] = simul
  }
  var.simul = matrix(0,Ny*365,N)
  if ( name.var == "Rain" ) {
    var.obs = data.obs[,4]
    for ( j in 1:N ) {
       var.simul[,j] = data.simul[,(5*(j-1)+1)]
    }
  }
  if ( name.var == "Tmin" ) {
    var.obs = data.obs[,5]
    for ( j in 1:N ) {
      var.simul[,j] = data.simul[,(5*(j-1)+2)]
     }
```

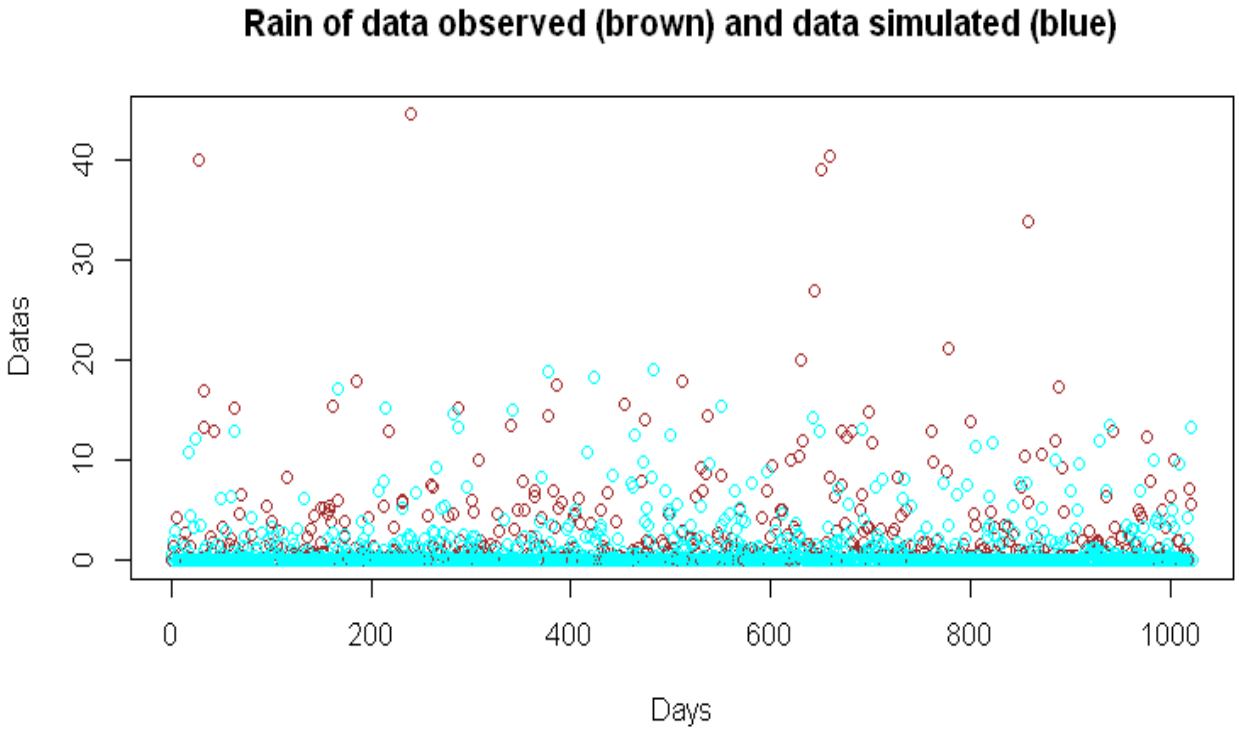
```
}
if ( name.var == "Tmax" ) {
  var.obs = data.obs[,6]
  if ( Tminmax == TRUE ) {
    var.obs = data.obs[,5] + data.obs[,6]
     for ( j in 1:N ) {
      var.simul[,j] = data.simul[,(5*(j-1)+3)]  + data.simul[,(5*(j-1)+2)]
    }
  }else{
    for ( j in 1:N ) {
      var.simul[,j] = data.simul[,(5*(j-1)+3)]
    }
  }
}
if ( name.var == "Radiation" ) {
  var.obs = data.obs[,7]
   for ( j in 1:N ) {
    var.simul[,j] = data.simul[,(5*(j-1)+4)]
   }
}
if ( name.var == "Wind" ) {
  var.obs = data.obs[,8]
   for ( j in 1:N ) {
     var.simul[,j] = data.simul[,5*j]
   }
}
observed = cbind(day,var.obs)
observed = as.data.frame(observed)
simulated = cbind(day,var.simul)
simulated = as.data.frame(simulated)
#####################################################################
if (stat == "mean") {
  mean.observed = tapply(observed$var.obs,list(observed$day),mean)
  mean.simulated = matrix(0,365,N)
  for ( i in 2:(N+1) ) {
      mean.simulated[,(i-1)] = tapply(simulated[,i],list(simulated$day),mean)
  }
  table.test = cbind(mean.observed,mean.simulated)
  table.test = as.data.frame(table.test)
  max.var = NULL
  min.var = NULL
  number = 0
  for ( i in 1:nrow(table.test) ) {
    max.var[i] = max(table.test[i,2:ncol(table.test)])
    min.var[i] = min(table.test[i,2:ncol(table.test)])
    if ( max.var[i] < table.test[i,1] | min.var[i] > table.test[i,1] ) {
      number = number+1
    }else {
      number = number
    }
  }
  par (mfrow=c(1,1))
  plot(mean.observed,ylab="means",xlab="days",col="red",type="l",main=paste("Confidence intervals for means",name.var,"to",alp
  for ( i in 1:N ) {
    lines(mean.simulated[,i],col="grey",type="l")
  }
  lines(max.var,col="black")
  lines(min.var,col="black")
  lines(mean.observed,col="red")
  if (number > 365*alpha) {
    print (paste("Means observed aren't in confidence intervals for",number,"days for",alpha,"level "))
  }else{
```

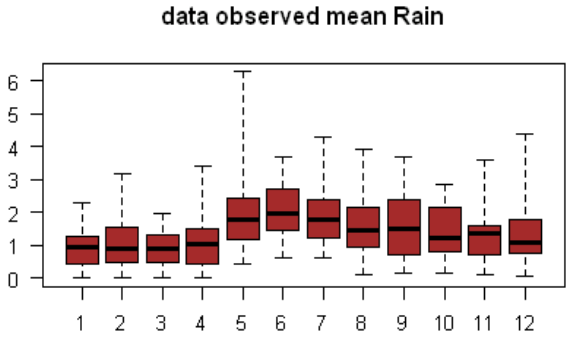
```
    print ("Every means observed are in confidence intervals for",alpha,"level.")
  }
}
if (stat == "sd") {
  sd.observed = tapply(observed$var.obs,list(observed$day),sd)
  sd.simulated = matrix(0,365,N)
  for ( i in 2:(N+1) ) {
    sd.simulated[,(i-1)] = tapply(simulated[,i],list(simulated$day),sd)
  }
  table.test = cbind(sd.observed,sd.simulated)
  table.test = as.data.frame(table.test)
  max.var = NULL
  min.var = NULL
  number = 0
  for ( i in 1:nrow(table.test) ) {
    max.var[i] = max(table.test[i,2:ncol(table.test)])
    min.var[i] = min(table.test[i,2:ncol(table.test)])
    if ( max.var[i] < table.test[i,1] | min.var[i] > table.test[i,1] ) {
      number = number+1
    }else {
      number = number
    }
  }
  par (mfrow=c(1,1))
  plot(sd.observed,ylab="sd",xlab="days",col="red",type="l",main=paste("Confidence intervals for sd",name.var,"to",alpha,"leve
  for ( i in 1:N ) {
    lines(sd.simulated[,i],col="grey",type="l")
  }
  lines(max.var,col="black")
  lines(min.var,col="black")
  lines(sd.observed,col="red")
  if (number > 365*alpha) {
    print (paste("Sd observed aren't in confidence intervals for",number,"days for",alpha,"level" ))
  }else{
    print ("Every sd observed are in confidence intervals for",alpha,"level.")
  }
}
if (stat == "correlation") {
  var.obs1 = observed$var.obs[-1]
  var.obs2 = observed$var.obs[-length(observed$var.obs)]
  observed = cbind(day[-length(day)],var.obs1,var.obs2)
  observed = as.data.frame(observed)
  colnames(observed) = c("day","var.obs1","var.obs2")
  var.simul1 = simulated[-1,2:ncol(simulated)]
  var.simul2 = simulated[-nrow(simulated),2:ncol(simulated)]
  simulated1 = cbind(day[-length(day)],var.simul1)
  simulated1 = as.data.frame(simulated1)
  simulated2 = cbind(day[-length(day)],var.simul2)
  simulated2 = as.data.frame(simulated2)
  cor.simulated = matrix(0,365,N)
  for ( i in 2:(N+1) ) {
    for ( j in unique(day)) {
      cor.simulated[j,(i-1)] = cor(simulated1[,i][simulated1$day == j],simulated2[,i][simulated2$day == j])
      }
  }
  cor.observed = NULL
  for ( j in unique(day)) {
    cor.observed[j] = cor(observed$var.obs1[observed$day == j],observed$var.obs2[observed$day == j])
  }
  table.test = cbind(cor.observed,cor.simulated)
  table.test = as.data.frame(table.test)
  max.var = NULL
```

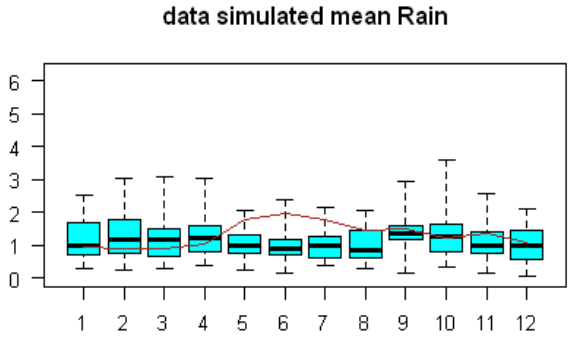
```
    min.var = NULL
    number = 0
    for ( i in 1:nrow(table.test) ) {
      max.var[i] = max(table.test[i,2:ncol(table.test)])
      min.var[i] = min(table.test[i,2:ncol(table.test)])
      if ( max.var[i] < table.test[i,1] | min.var[i] > table.test[i,1] ) {
        number = number+1
      }else {
        number = number
      }
    }
    par (mfrow=c(1,1))
    plot(cor.observed,ylab="correlation",xlab="days",col="red",type="l",main=paste("Confidence intervals for correlation",name.v
    for ( i in 1:N ) {
      lines(cor.simulated[,i],col="grey",type="l")
    }
    lines(max.var,col="black")
    lines(min.var,col="black")
    lines(cor.observed,col="red")
    if (number > 365*alpha) {
      print (paste("Correlation observed aren't in confidence intervals for",number,"days for",alpha,"level" ))
    }else{
      print ("Every correlation observed are in confidence intervals for",alpha,"level.")
    }
  }
}
```

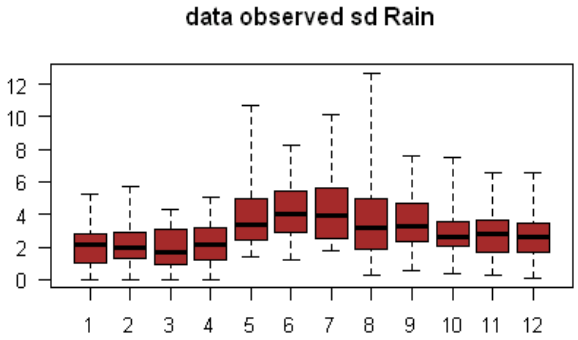
5.3.2 Comparaison données simulées avec WACSgen et données récoltées sur le site de Colmar - clustering = soft - Nclusters = 2

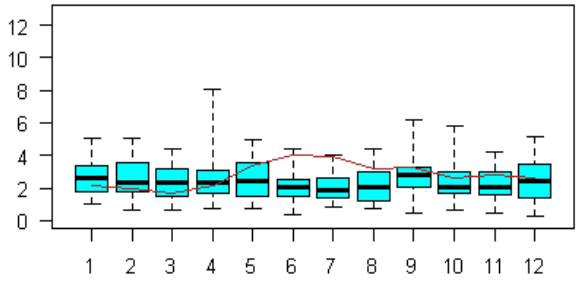
Comparaison variable par variable

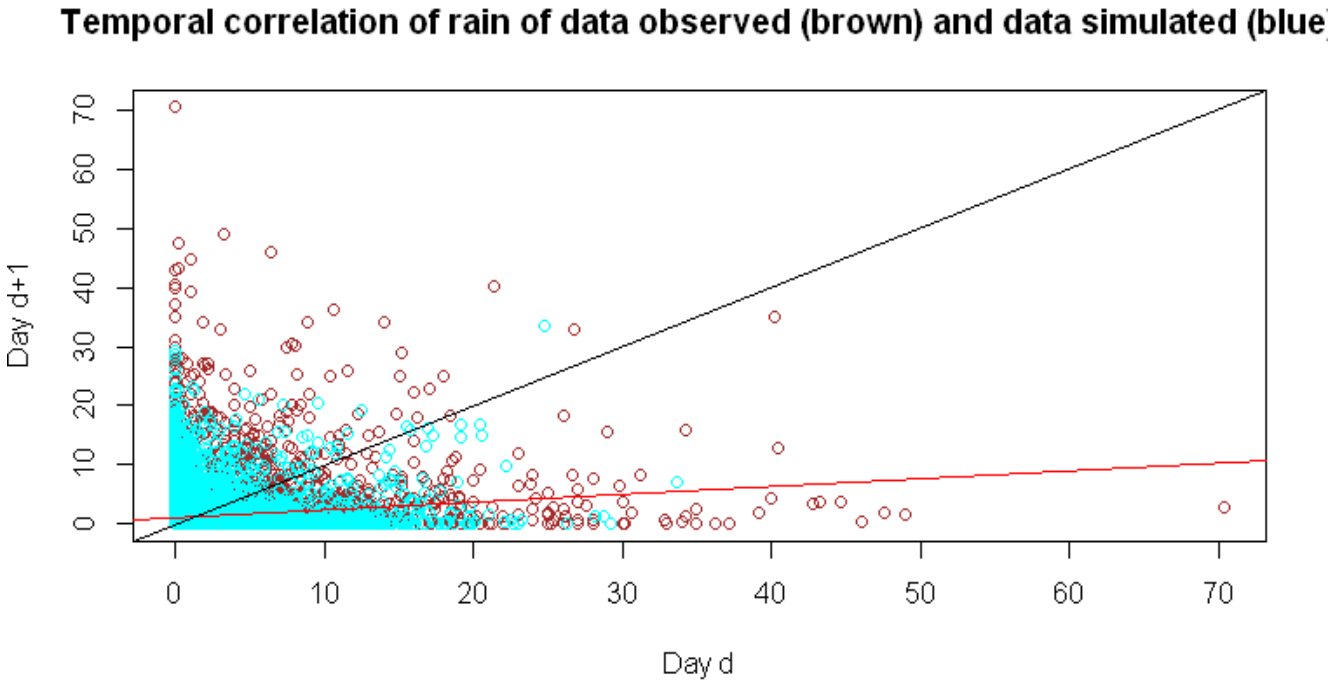


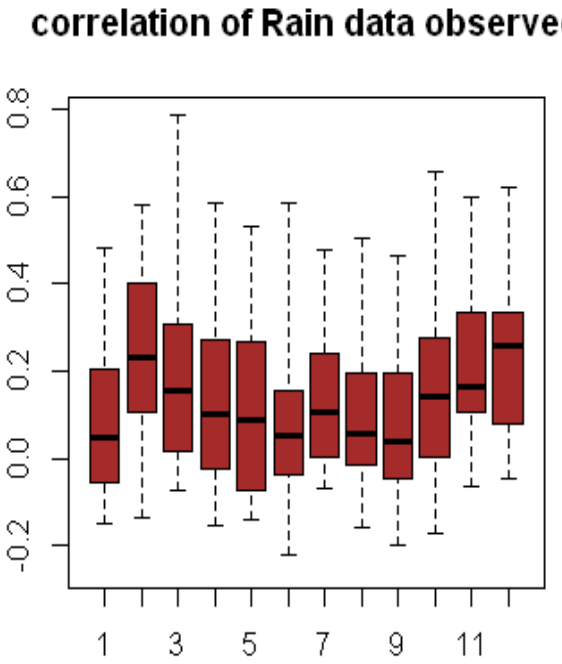


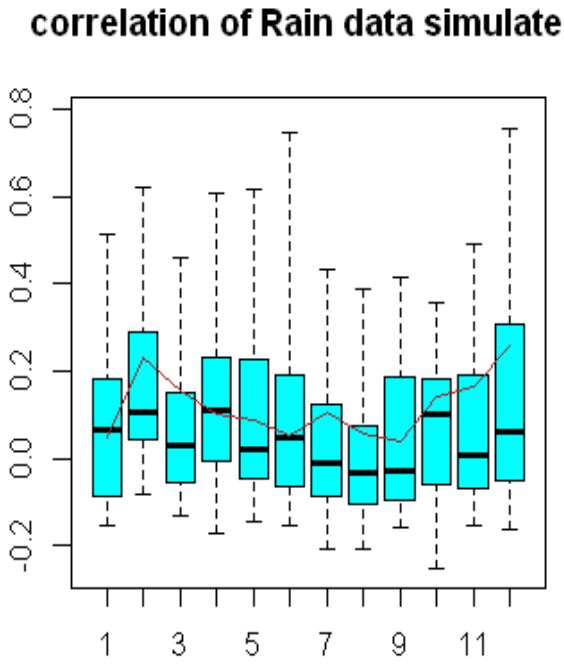






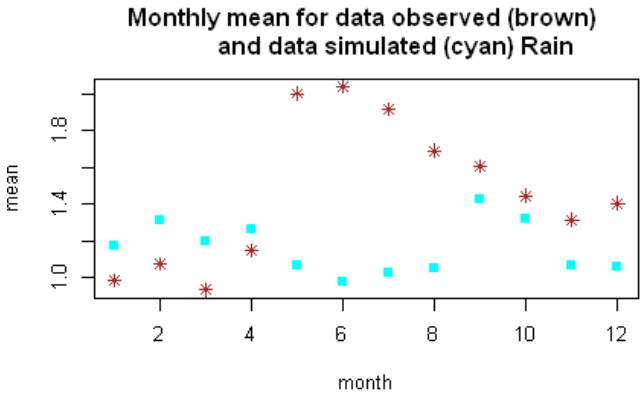


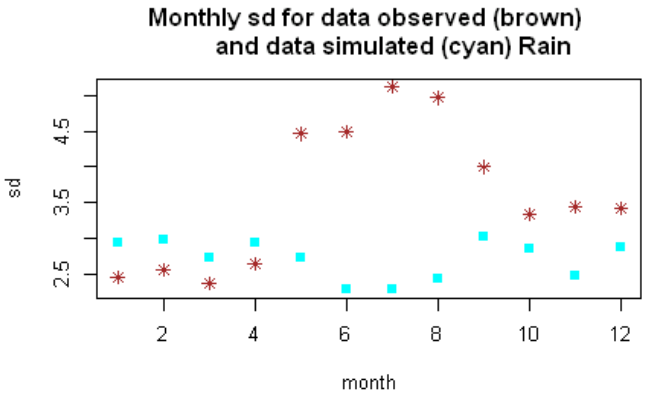


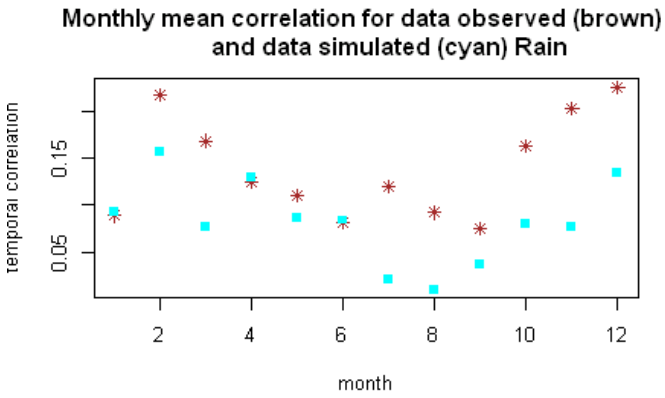


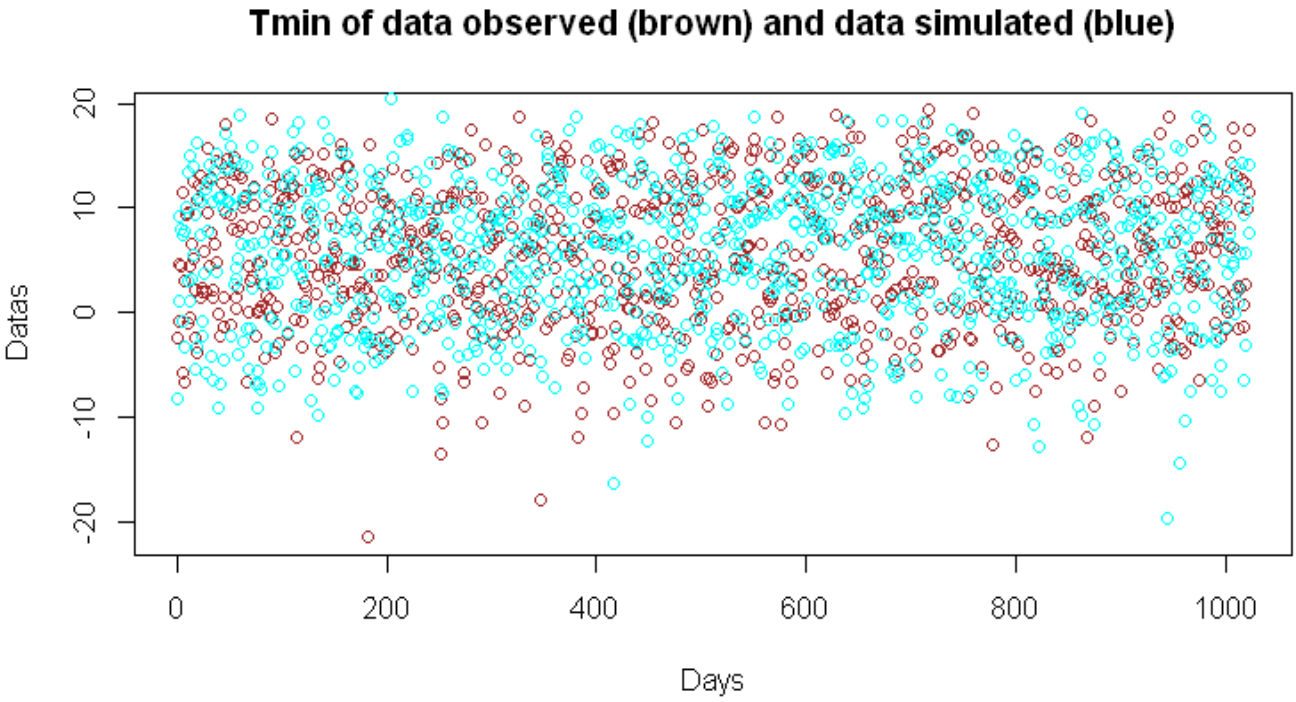

```
[1] "summary of data observed correlation Rain :"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
0.03768 0.05486 0.09750 0.11360 0.14840 0.25930 4.00000
[1] "summary of data simulated correlation Rain :"
     Min.   1st Qu.    Median      Mean   3rd Qu.  Max.
-0.033930  0.005001  0.039000  0.040890  0.075530  0.110500
```

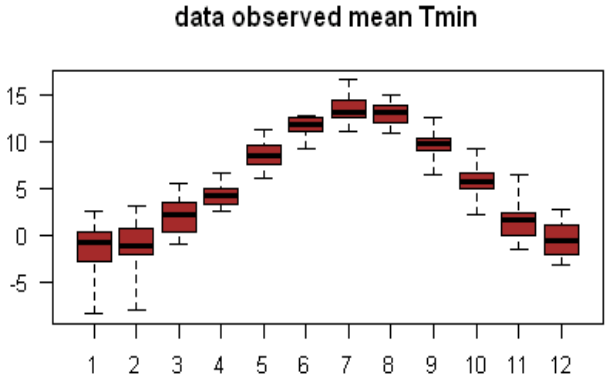
```
[1] "Global mean for data observed Rain is 1.466 and for data simulated is 1.161"
[1] "Global sd for data observed Rain is 3.75 and for data simulated is 2.716"
[1] "Global correlation for data observed Rain is 0.18 and for data simulated is 0.133"
```

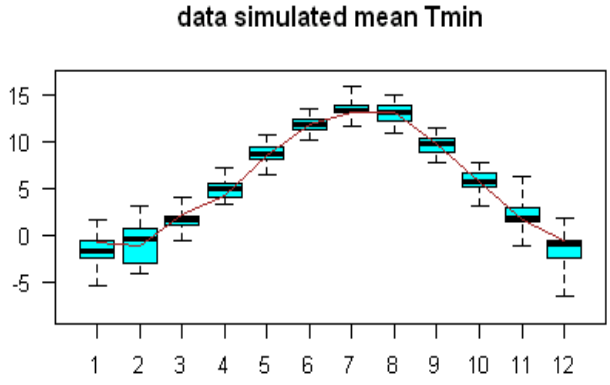


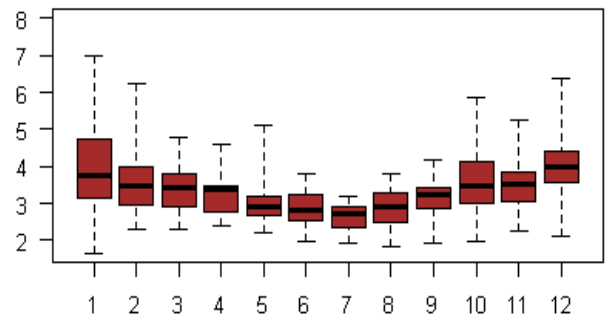


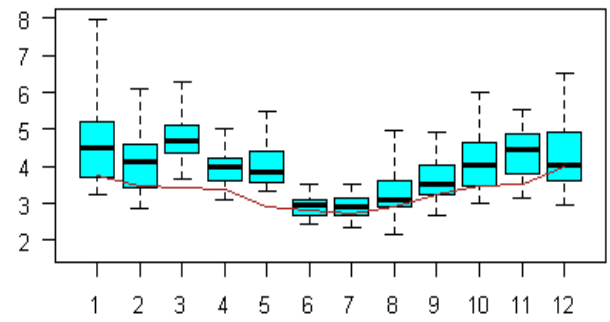



Tmin of data observed (brown) and data simulated (blue)
Datas
Days
20
10
0
-10
-20
0
200
400
600
800
1000

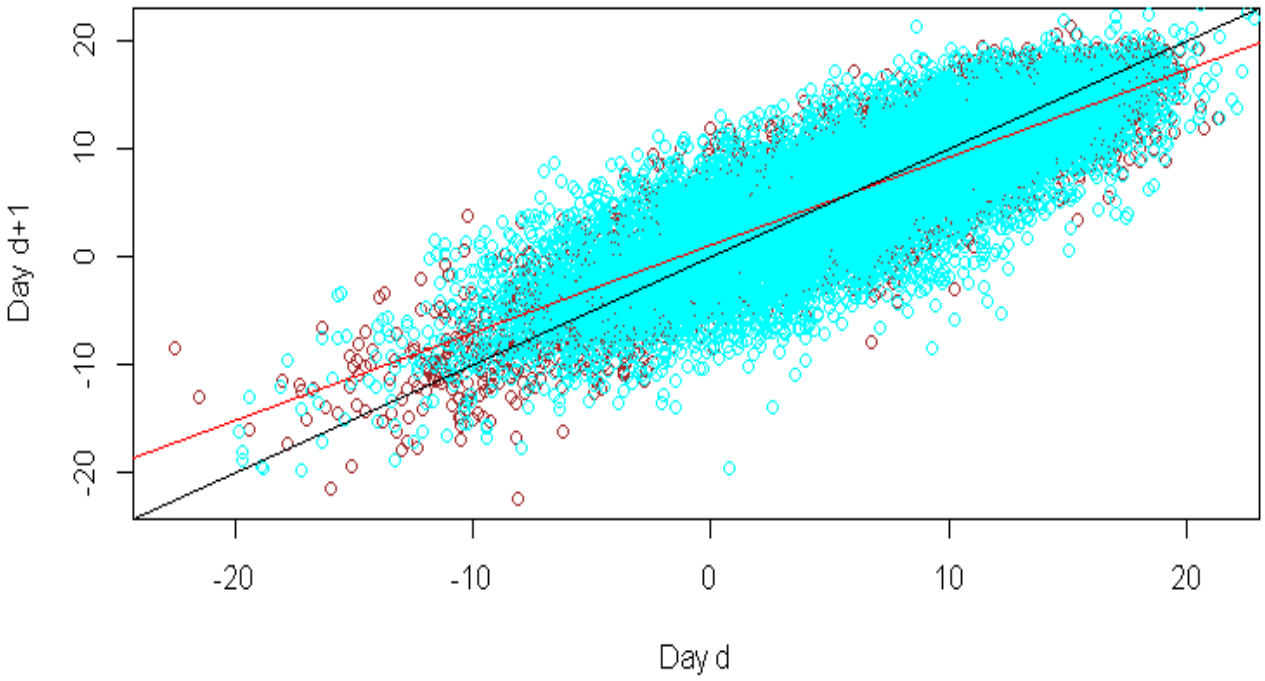

data observed mean Tmin
15
10
5
0
-5
1 2 3 4 5 6 7 8 9 10 11 12


data simulated mean Tmin
15
10
5
0
-5
1 2 3 4 5 6 7 8 9 10 11 12

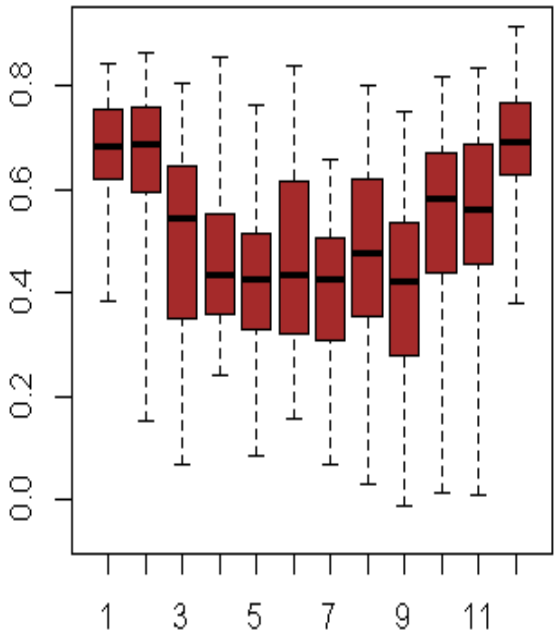

data observed sd Tmin
8
7
6
5
4
3
2
1 2 3 4 5 6 7 8 9 10 11 12


data simulated sd Tmin
8
7
6
5
4
3
2
1 2 3 4 5 6 7 8 9 10 11 12

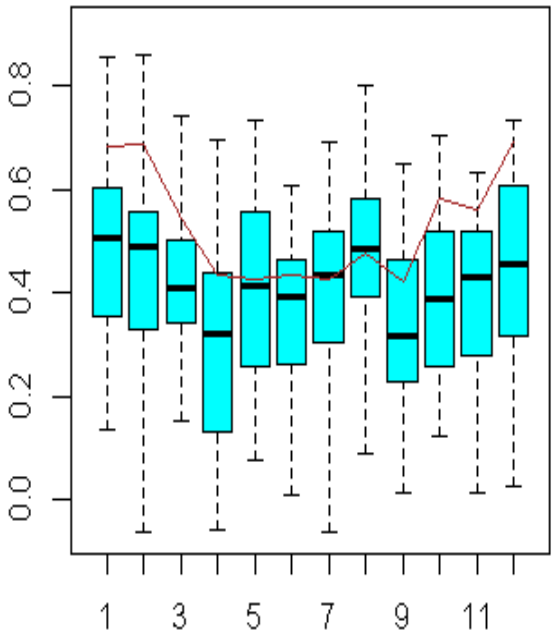
Temporal correlation of Tmin of data observed (brown) and data simulated (blue



correlation of Tmin data observed

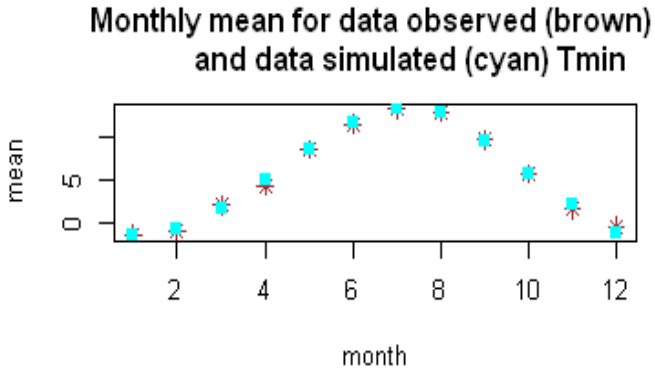


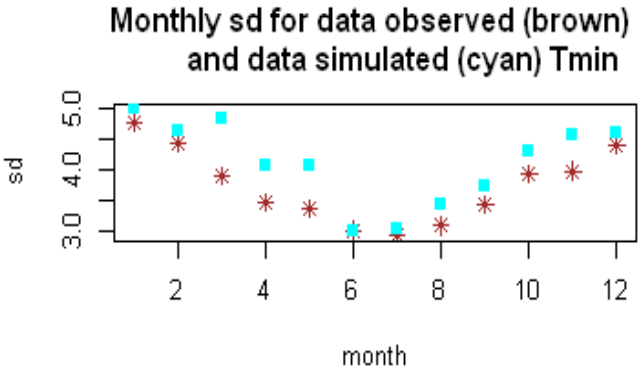
correlation of Tmin data simulated

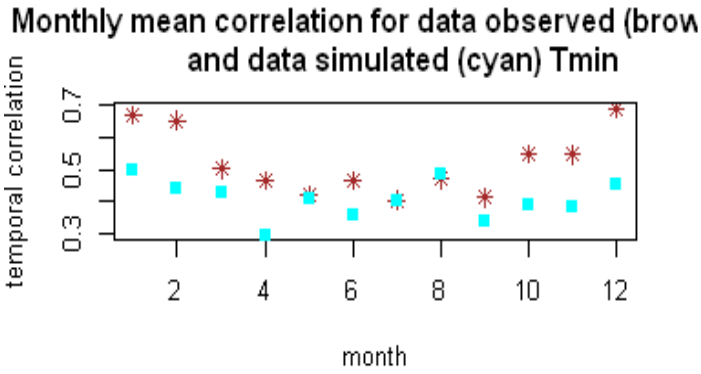


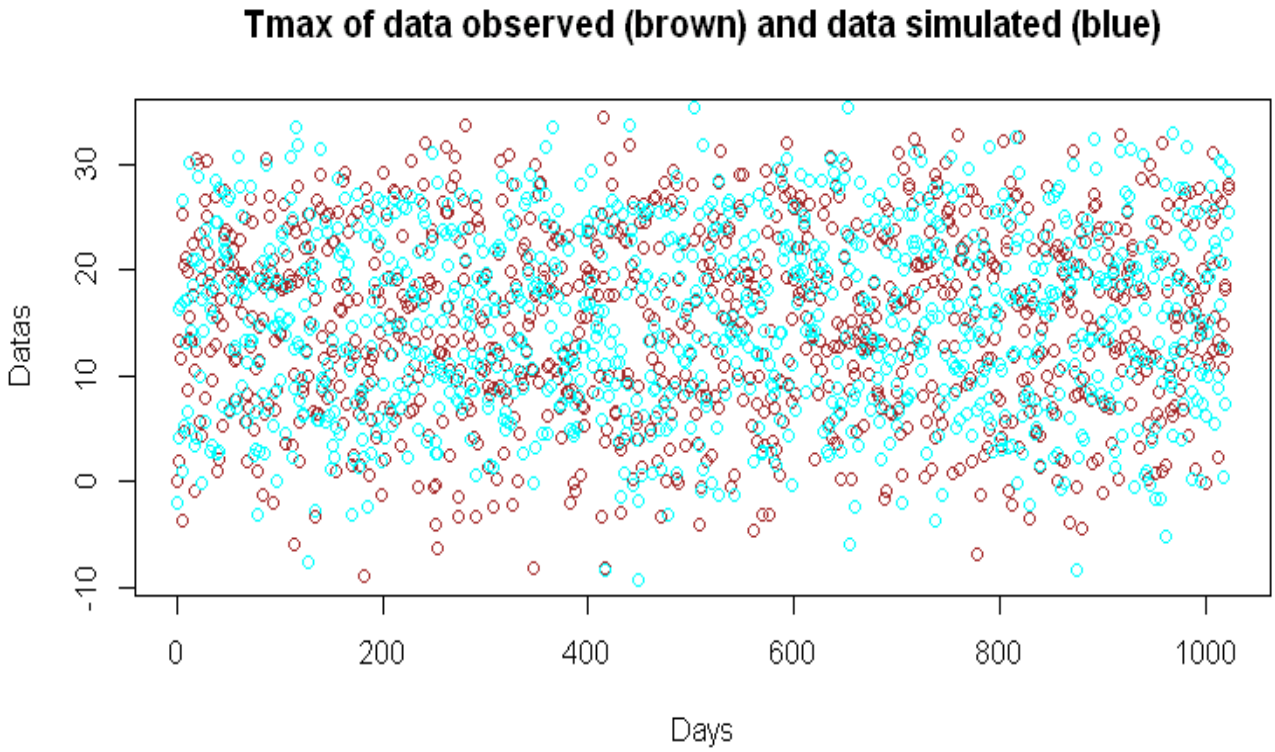
```
[1] "summary of data observed correlation Tmin :"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.4215  0.4316  0.5111  0.5310  0.6068  0.6923
[1] "summary of data simulated correlation Tmin :"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.3167  0.3931  0.4218  0.4209  0.4647  0.5059
```

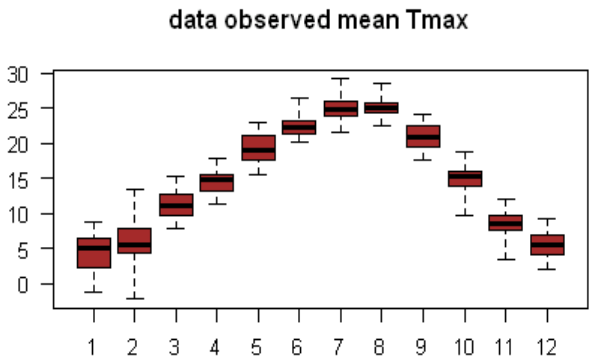
```
[1] "Global mean for data observed Tmin is 5.612 and for data simulated is 5.605"
[1] "Global sd for data observed Tmin is 6.534 and for data simulated is 6.821"
[1] "Global correlation for data observed Tmin is 0.886 and for data simulated is 0.814"
```

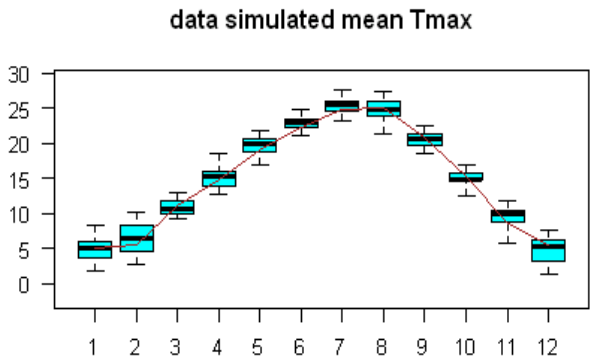


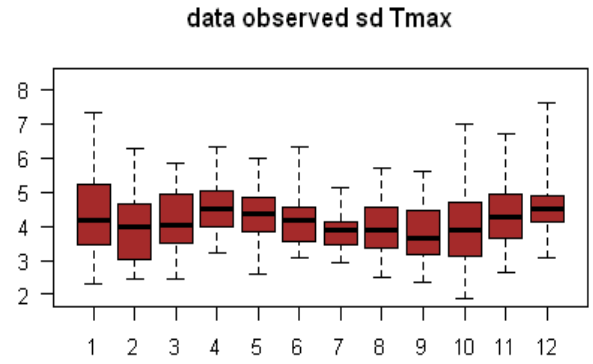


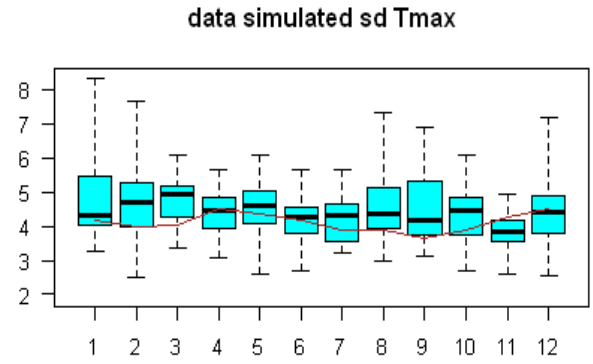


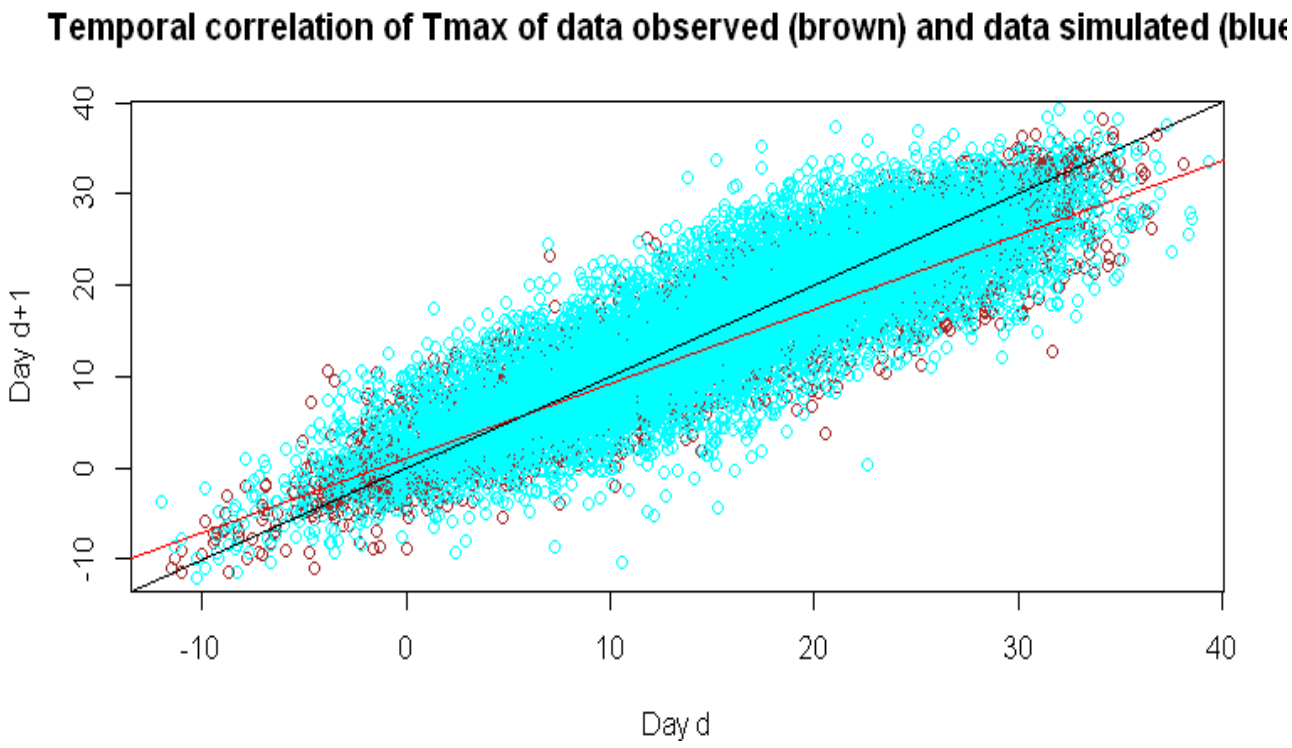

Tmax of data observed (brown) and data simulated (blue)
Datas
-10
0
10
20
30
0
200
400
600
800
1000
Days

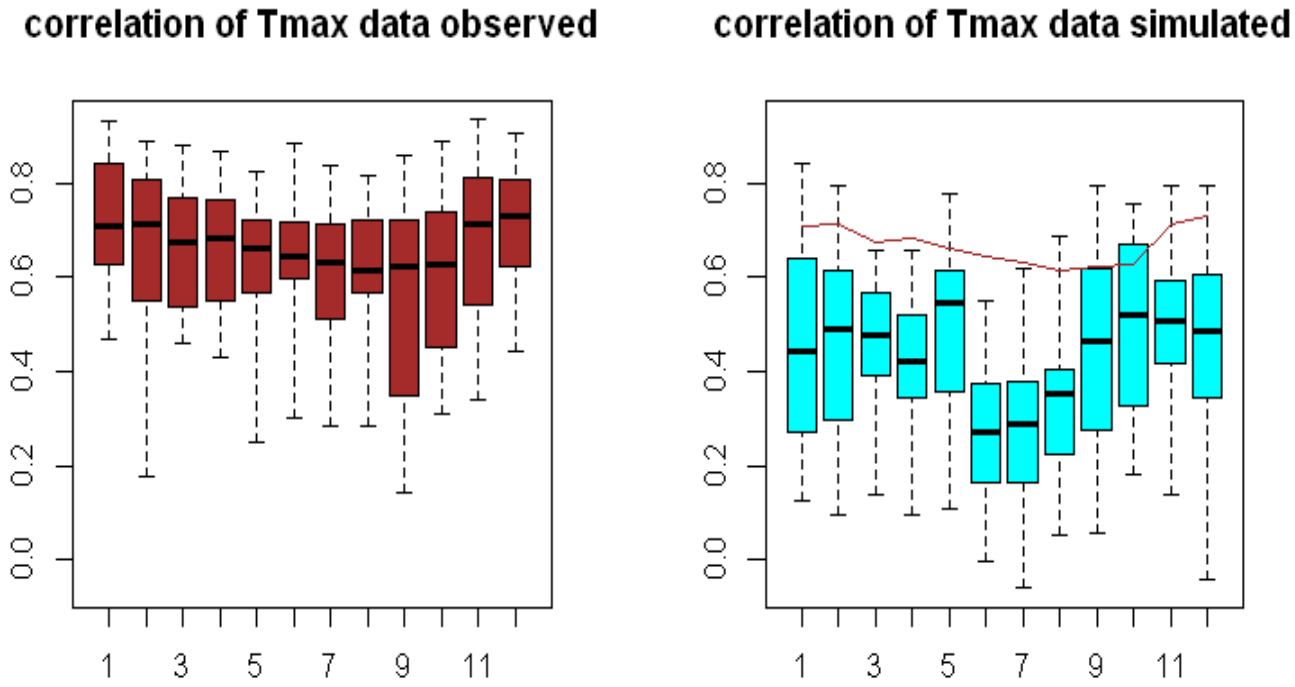

data observed mean Tmax
0
5
10
15
20
25
30
1
2
3
4
5
6
7
8
9
10
11
12


data simulated mean Tmax
0
5
10
15
20
25
30
1
2
3
4
5
6
7
8
9
10
11
12


data observed sd Tmax
2
3
4
5
6
7
8
1
2
3
4
5
6
7
8
9
10
11
12

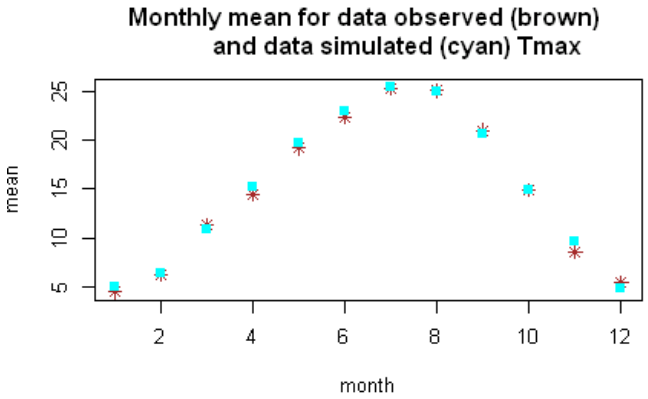

data simulated sd Tmax
2
3
4
5
6
7
8
1
2
3
4
5
6
7
8
9
10
11
12

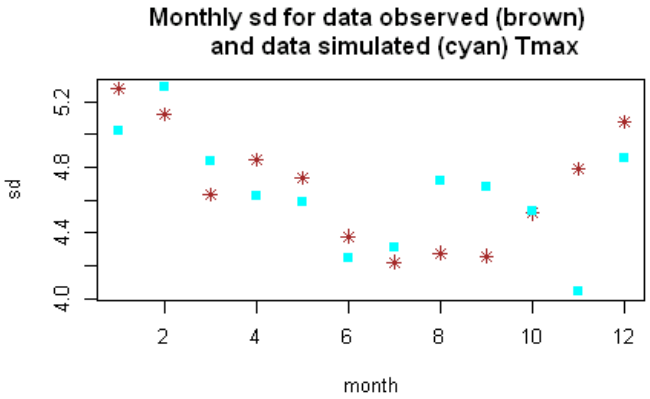


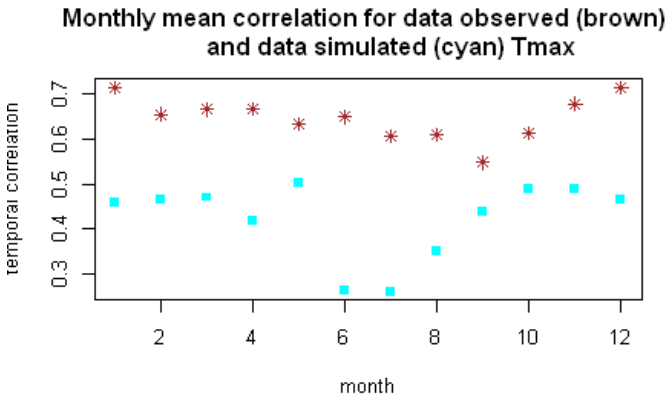


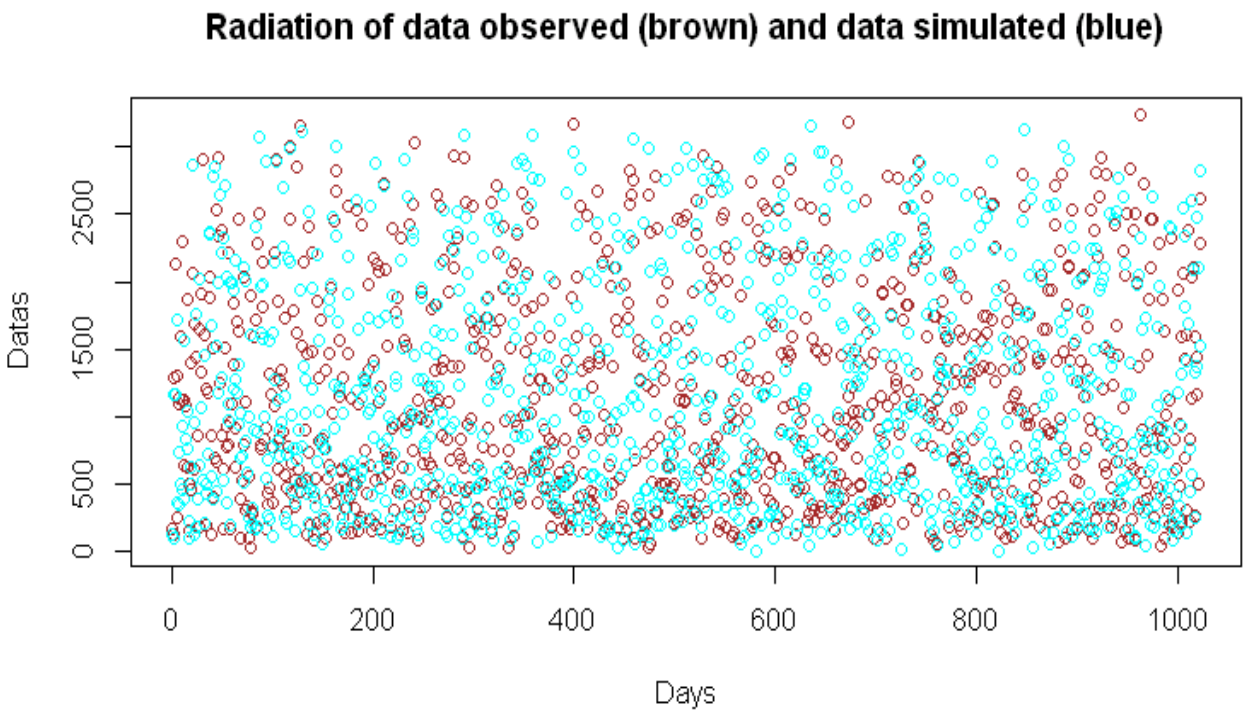

```
[1] "summary of data observed correlation Tmax :"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.6152  0.6302  0.6678  0.6686  0.7093  0.7290
[1] "summary of data simulated correlation Tmax :"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2728  0.4029  0.4718  0.4388  0.4926  0.5464
```

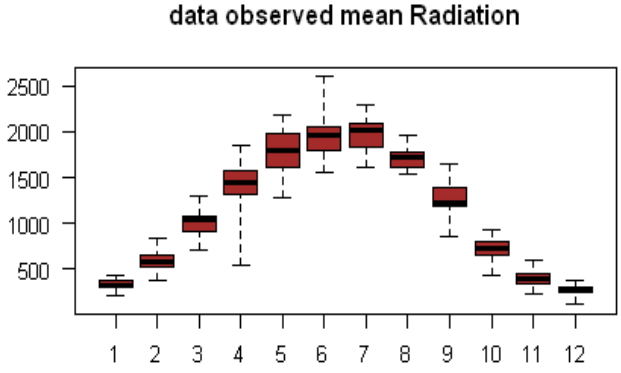
```
[1] "Global mean for data observed Tmax is 14.93 and for data simulated is 15.104"
[1] "Global sd for data observed Tmax is 8.701 and for data simulated is 8.678"
[1] "Global correlation for data observed Tmax is 0.927 and for data simulated is 0.859"
```

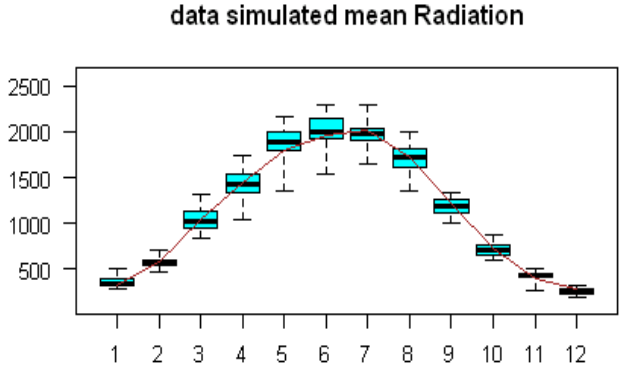


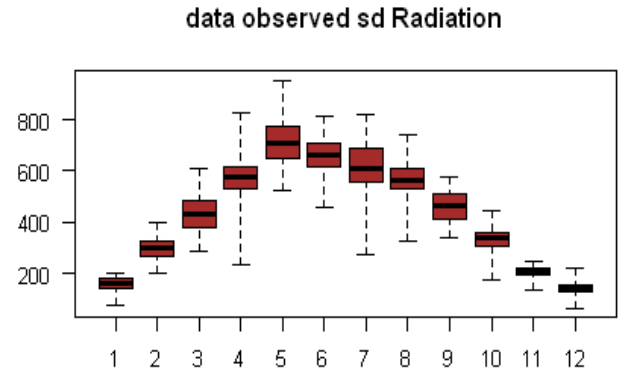


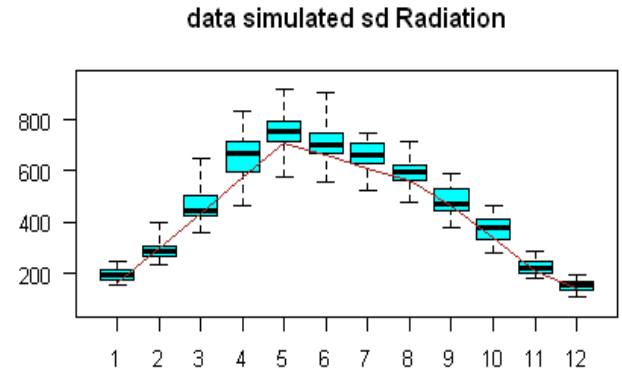


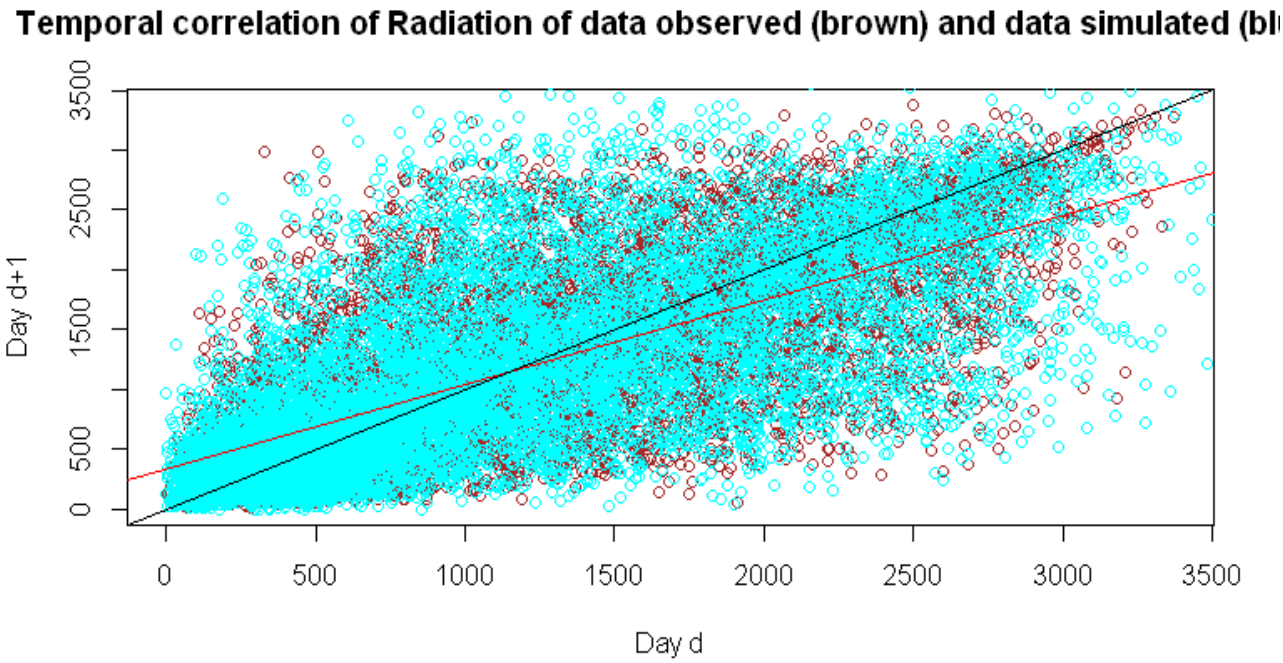

Radiation of data observed (brown) and data simulated (blue)
Datas
0
500
1500
2500
0
200
400
600
800
1000
Days

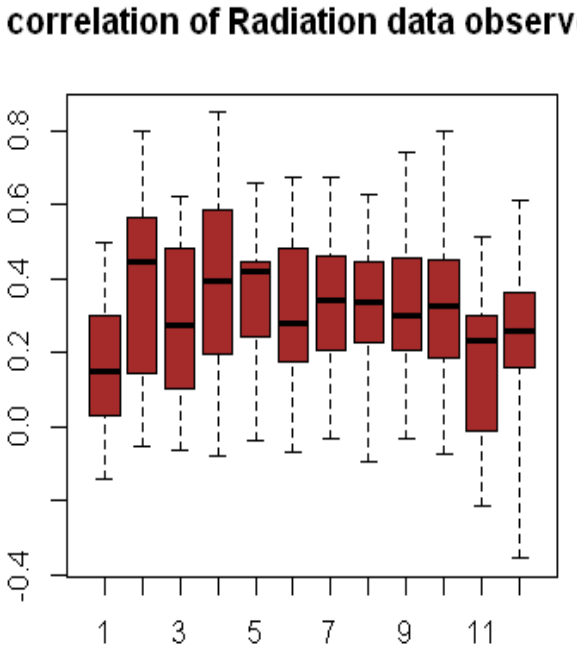

data observed mean Radiation
500
1000
1500
2000
2500
1
2
3
4
5
6
7
8
9
10
11
12

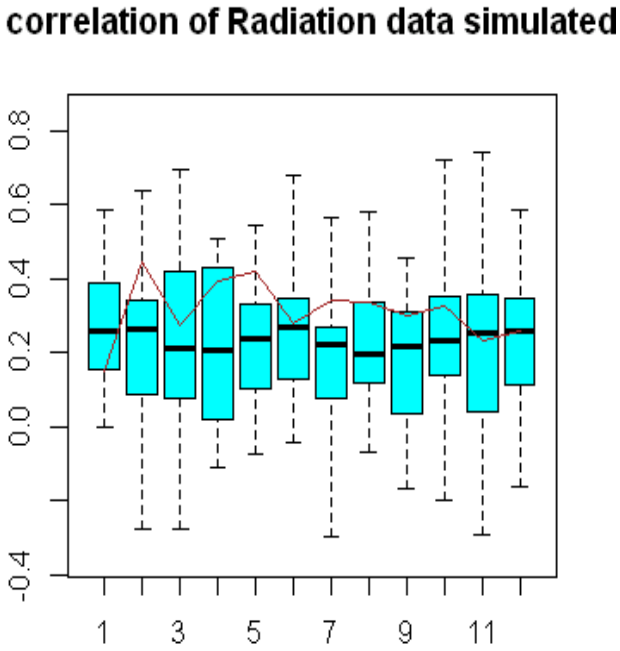

data simulated mean Radiation
500
1000
1500
2000
2500
1
2
3
4
5
6
7
8
9
10
11
12


data observed sd Radiation
200
400
600
800
1
2
3
4
5
6
7
8
9
10
11
12


data simulated sd Radiation
200
400
600
800
1
2
3
4
5
6
7
8
9
10
11
12

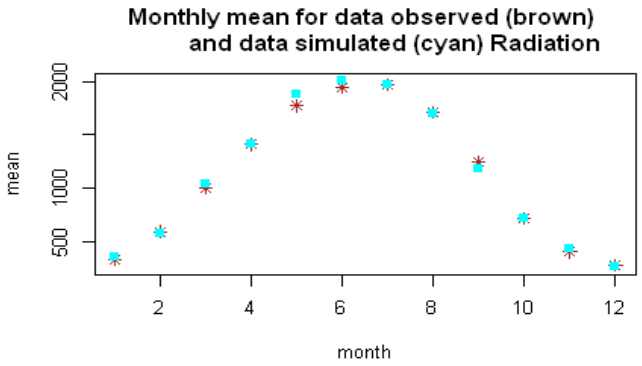


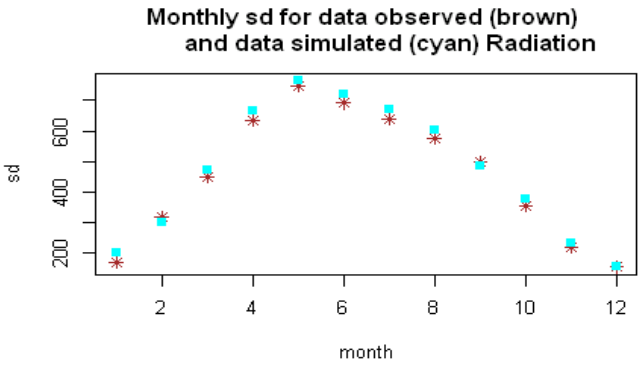


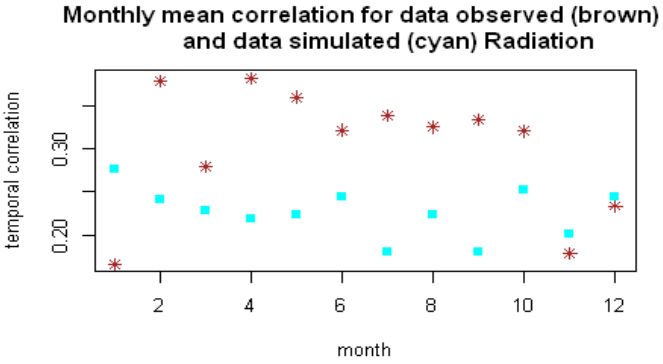


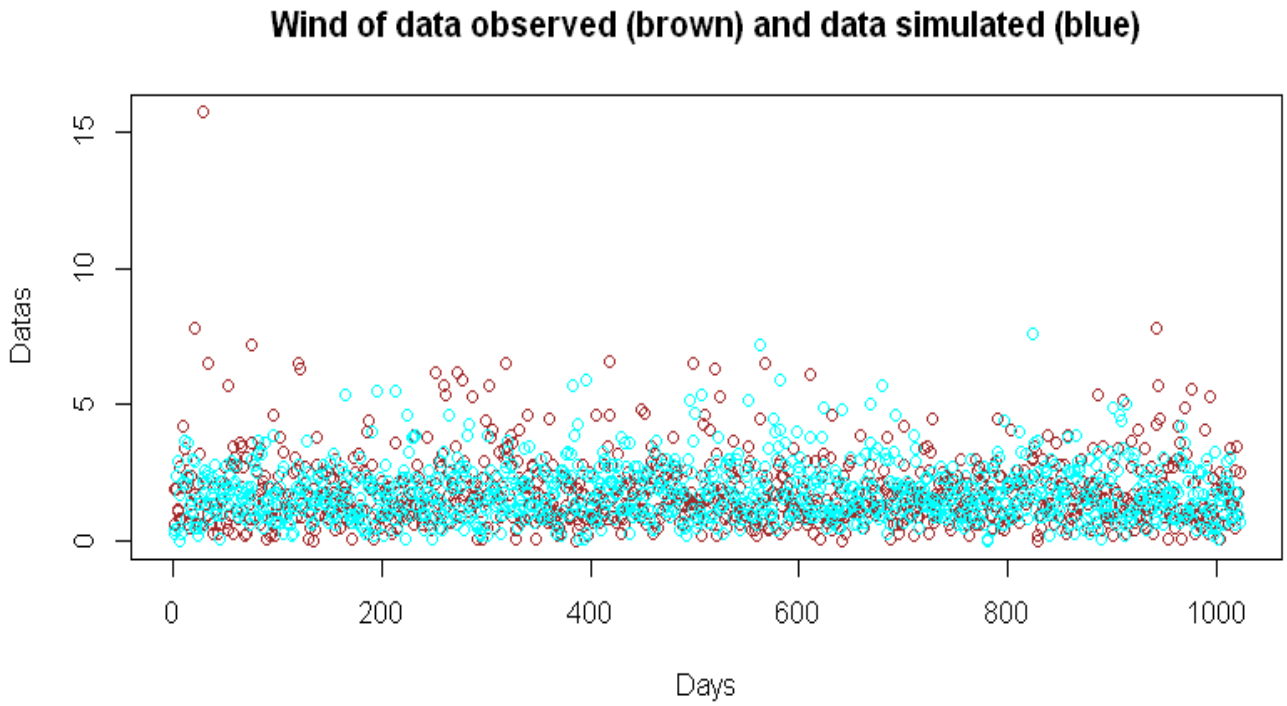

```
[1] "summary of data observed correlation Radiation :"
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.1507  0.2714  0.3131  0.3120  0.3525  0.4439
[1] "summary of data simulated correlation Radiation :"
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.1979  0.2154  0.2348  0.2355  0.2561  0.2701
```

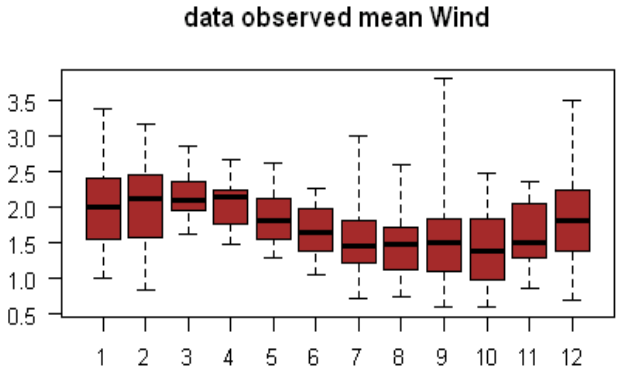
```
[1] "Global mean for data observed Radiation is 1117.682 and for data simulated is
1131.899"
[1] "Global sd for data observed Radiation is 793.705 and for data simulated is 815.053"
[1] "Global correlation for data observed Radiation is 0.774 and for data simulated is
0.709"
```

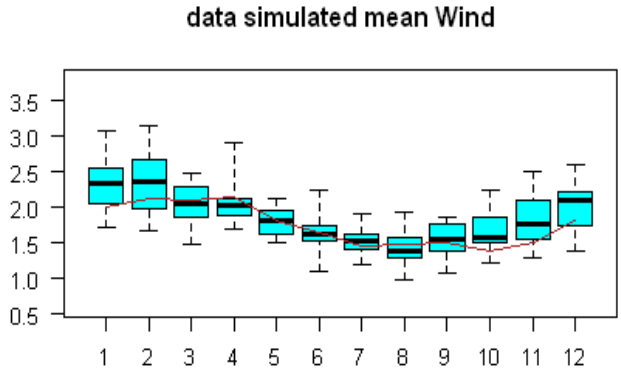


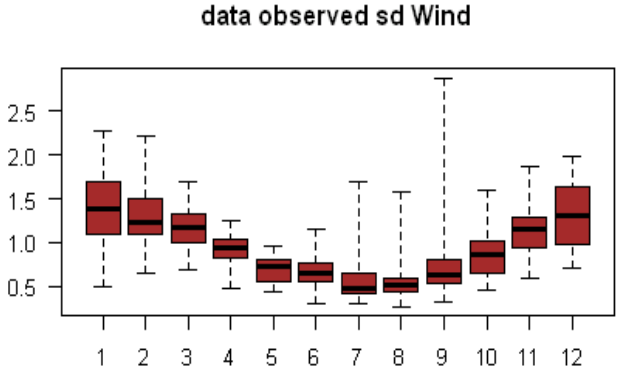


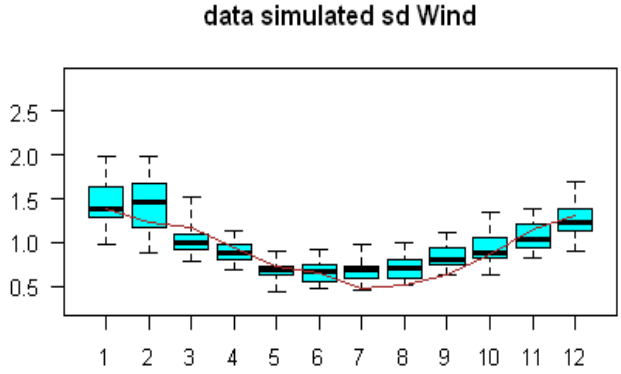


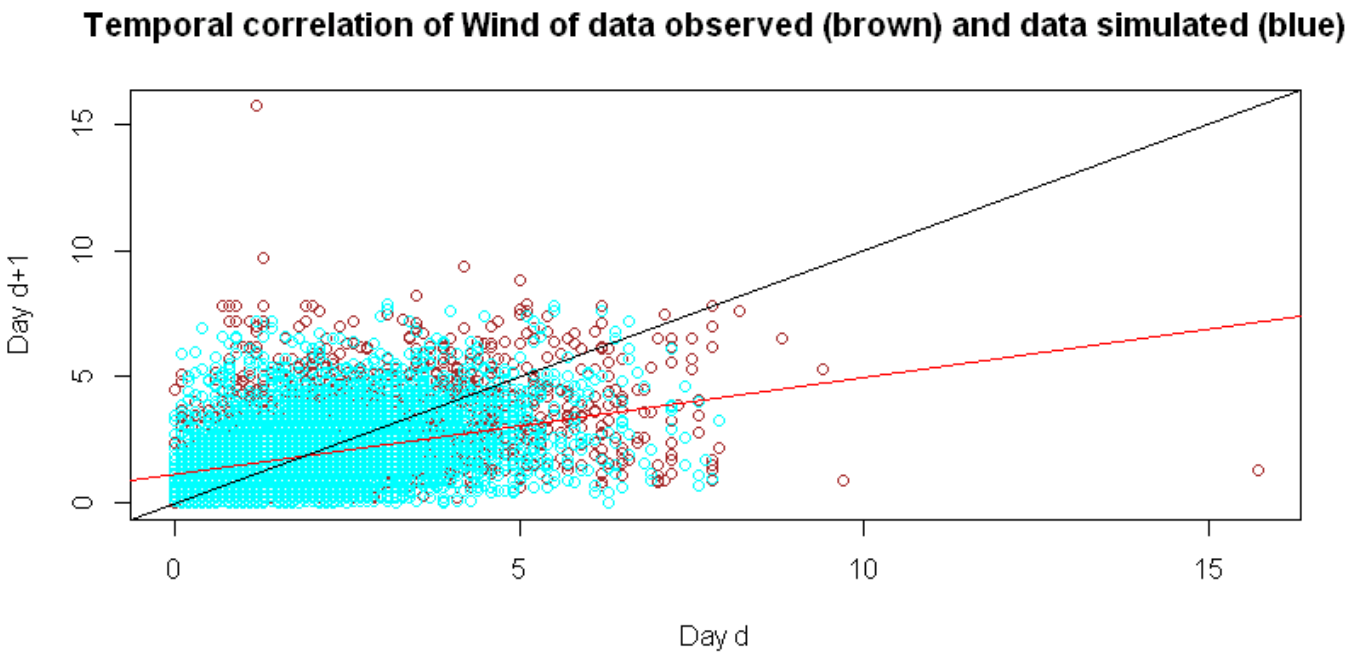

Wind of data observed (brown) and data simulated (blue)
Datas
0
5
10
15
0
200
400
600
800
1000
Days

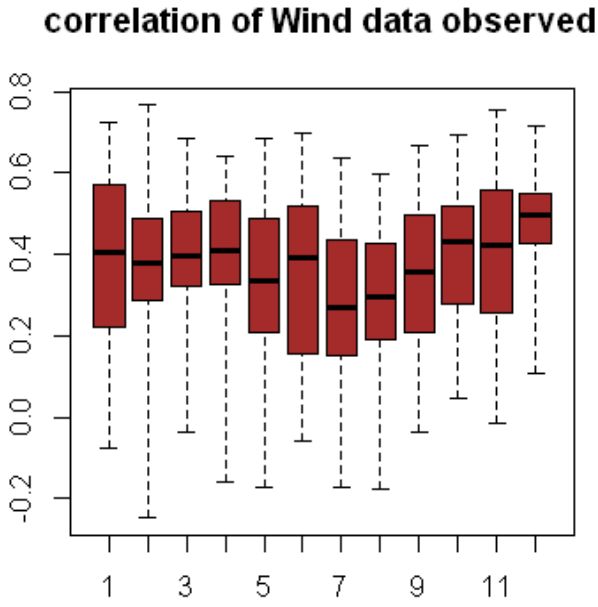

data observed mean Wind
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1
2
3
4
5
6
7
8
9
10
11
12

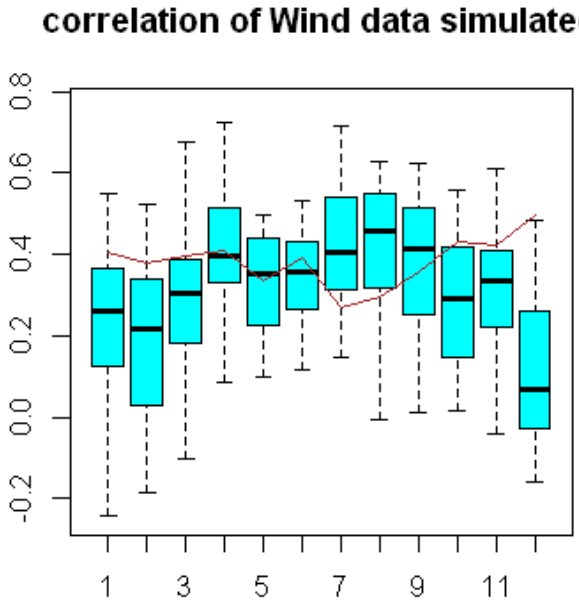

data simulated mean Wind
0.5
1.0
1.5
2.0
2.5
3.0
3.5
1
2
3
4
5
6
7
8
9
10
11
12


data observed sd Wind
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
8
9
10
11
12


data simulated sd Wind
0.5
1.0
1.5
2.0
2.5
1
2
3
4
5
6
7
8
9
10
11
12

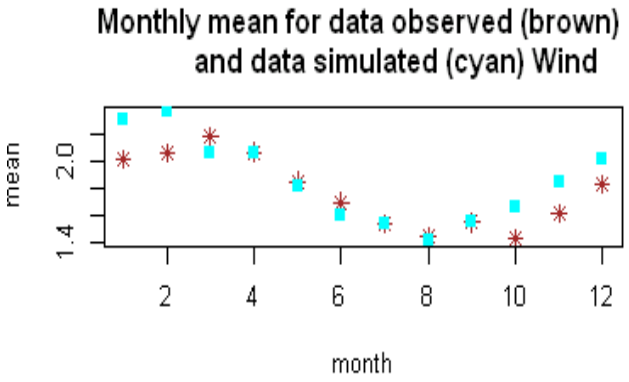


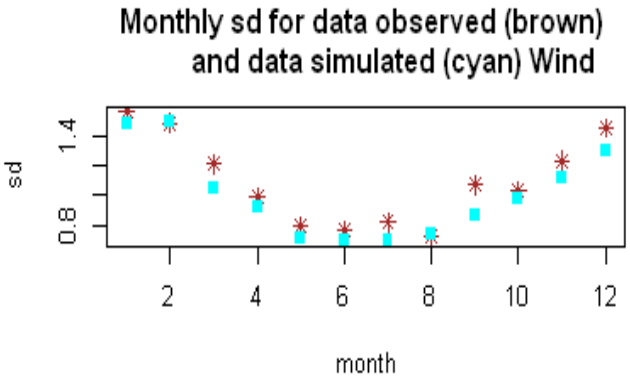


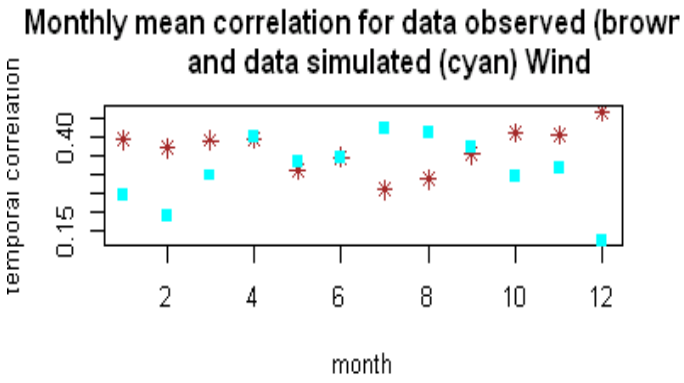



```
[1] "summary of data observed correlation Wind :"
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.2690  0.3513  0.3937  0.3823  0.4126  0.4975
[1] "summary of data simulated correlation Wind :"
    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
```

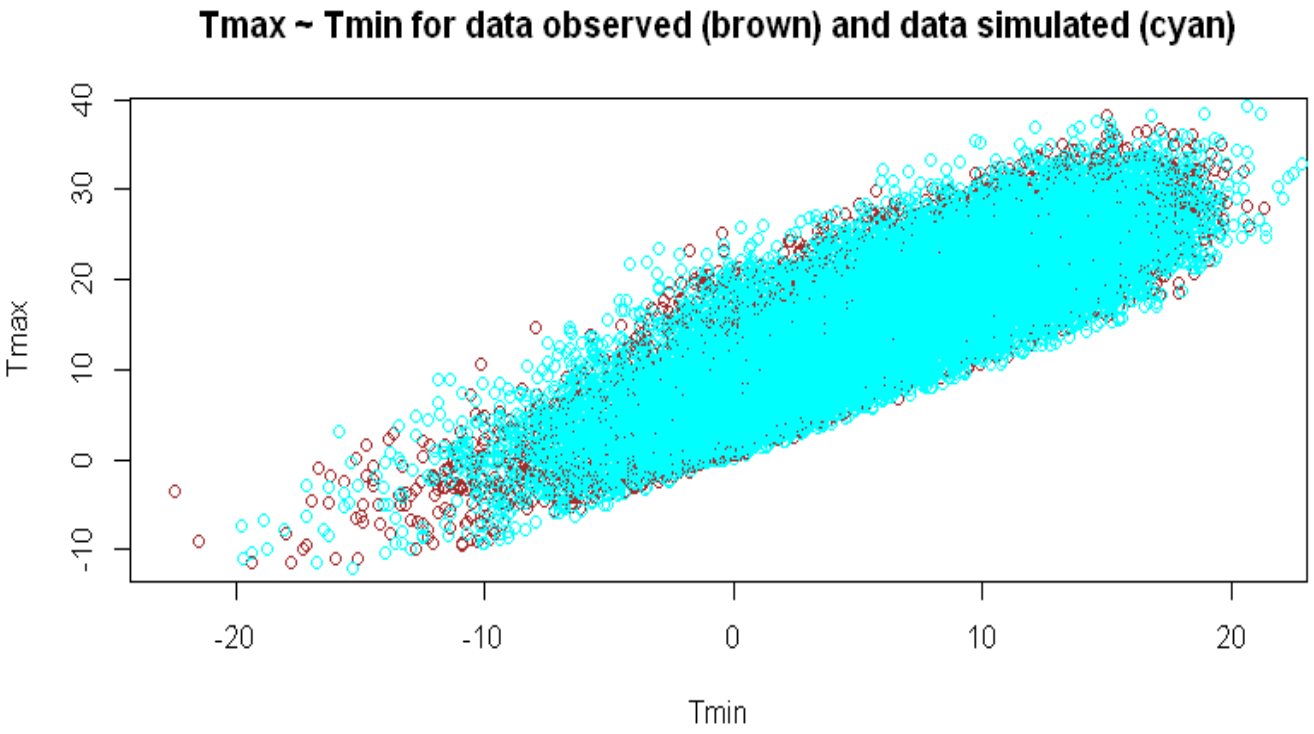
```
0.06651 0.28340 0.34330 0.32080 0.39650 0.45910

[1] "Global mean for data observed Wind is 1.771 and for data simulated is 1.847"
[1] "Global sd for data observed Wind is 1.157 and for data simulated is 1.081"
[1] "Global correlation for data observed Wind is 0.519 and for data simulated is 0.382"
```

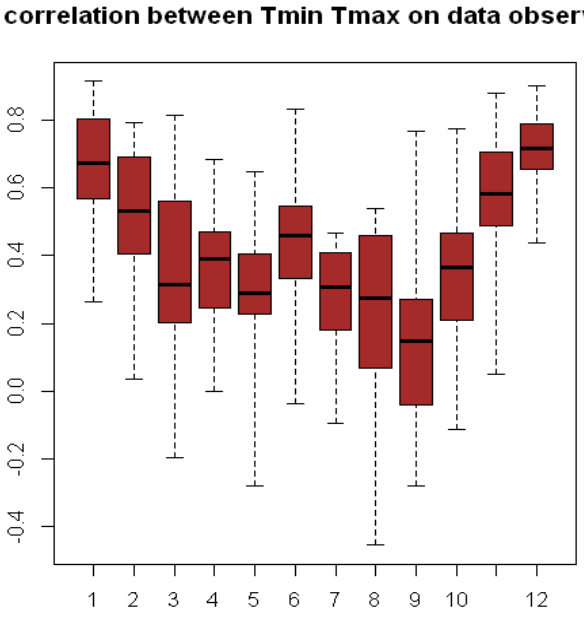


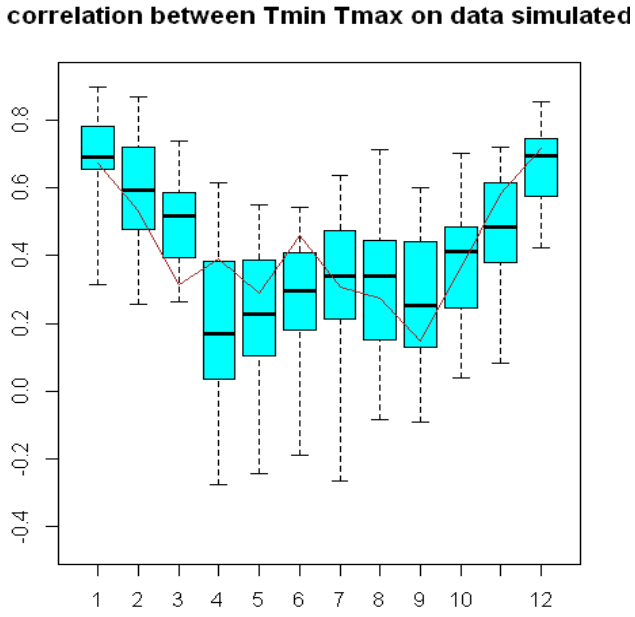


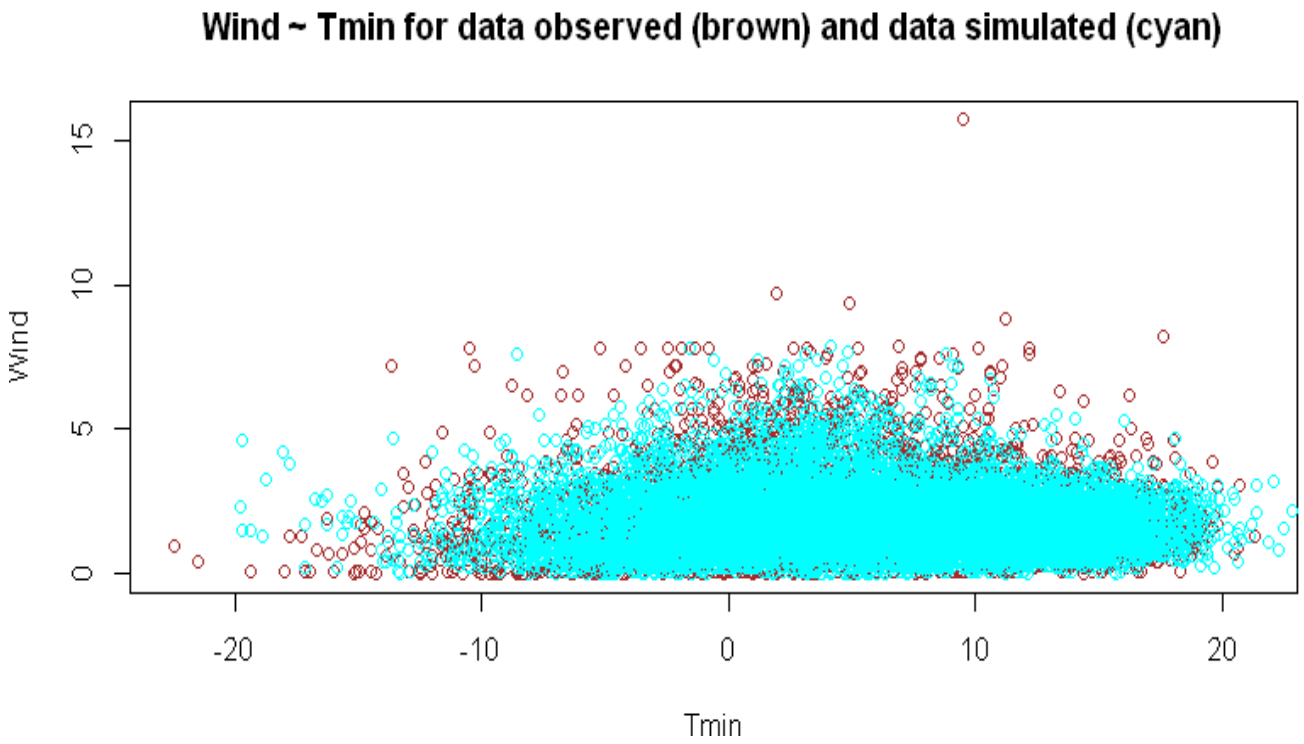


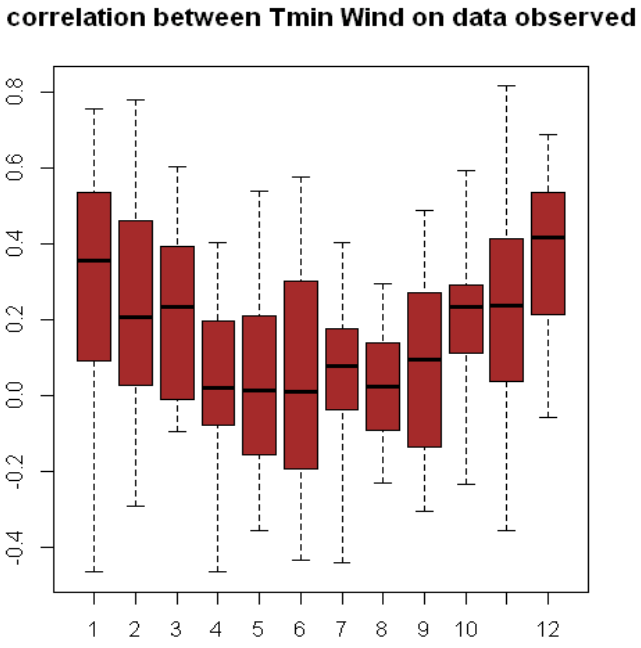
Comparaison bivariée

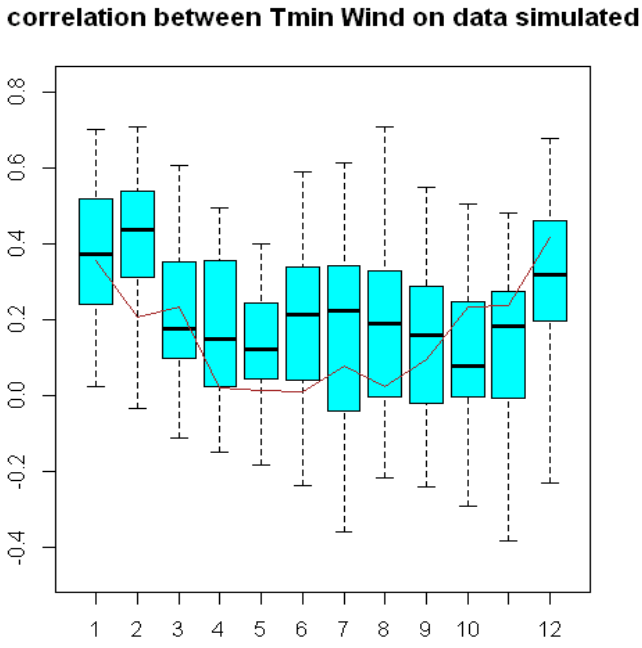


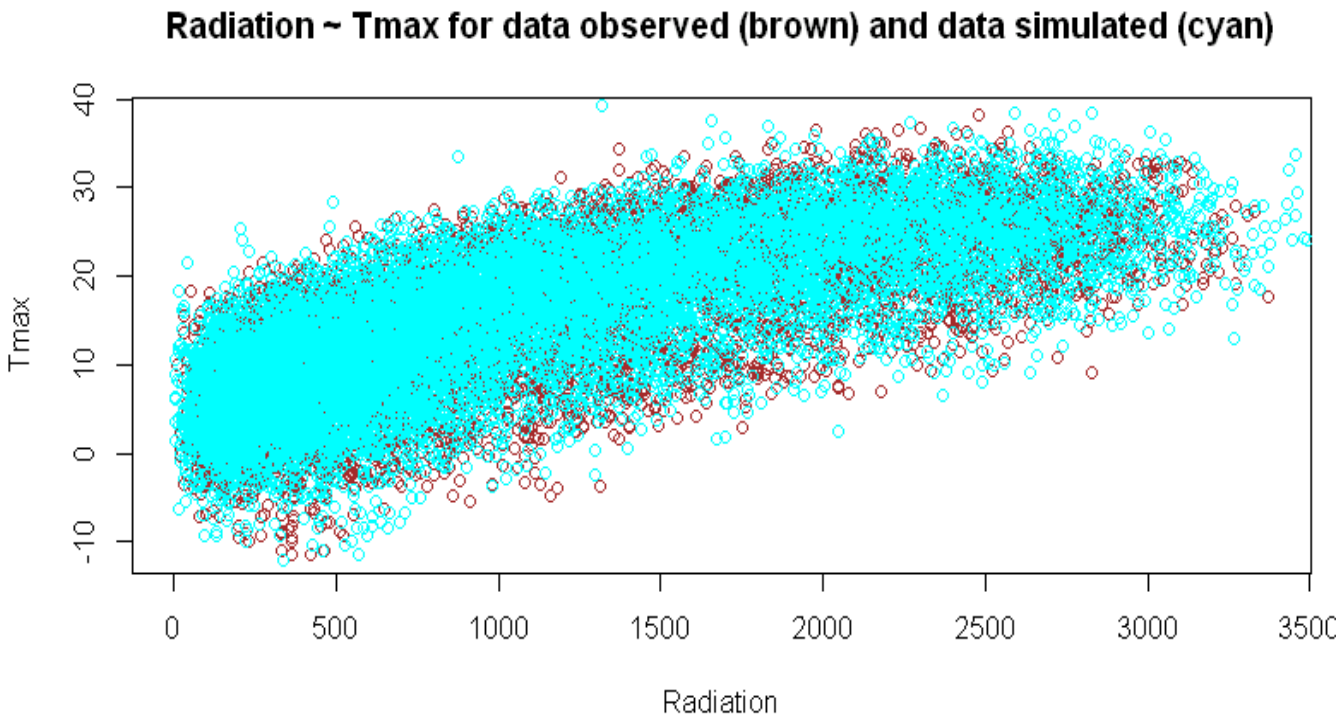


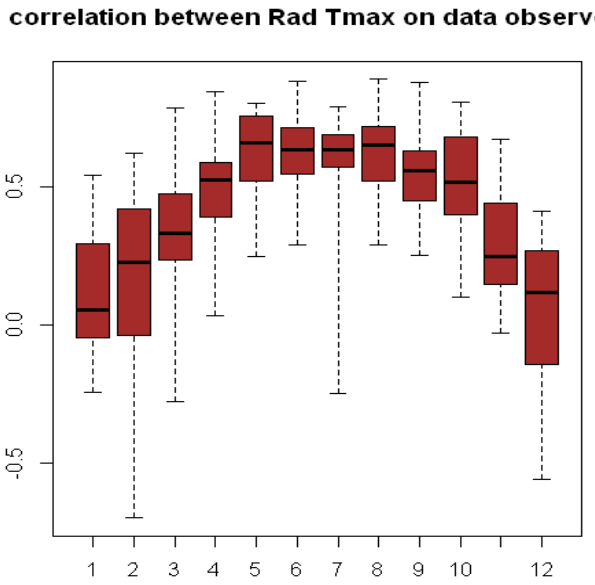


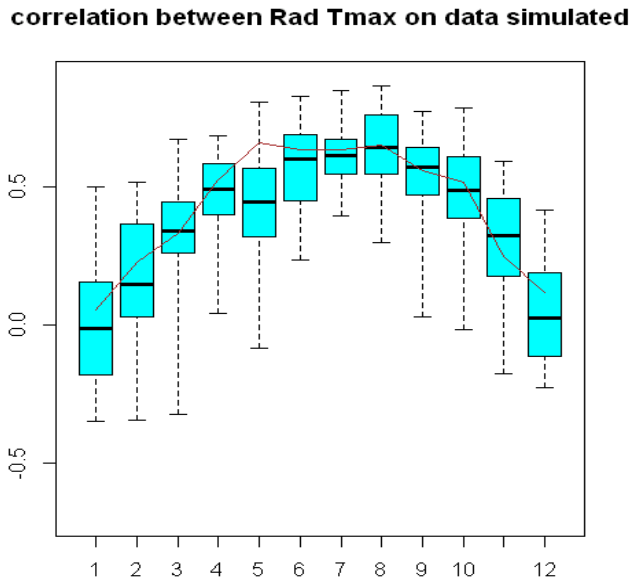

Wind ~ Tmin for data observed (brown) and data simulated (cyan)
Wind
0
5
10
15
-20
-10
0
10
20
Tmin

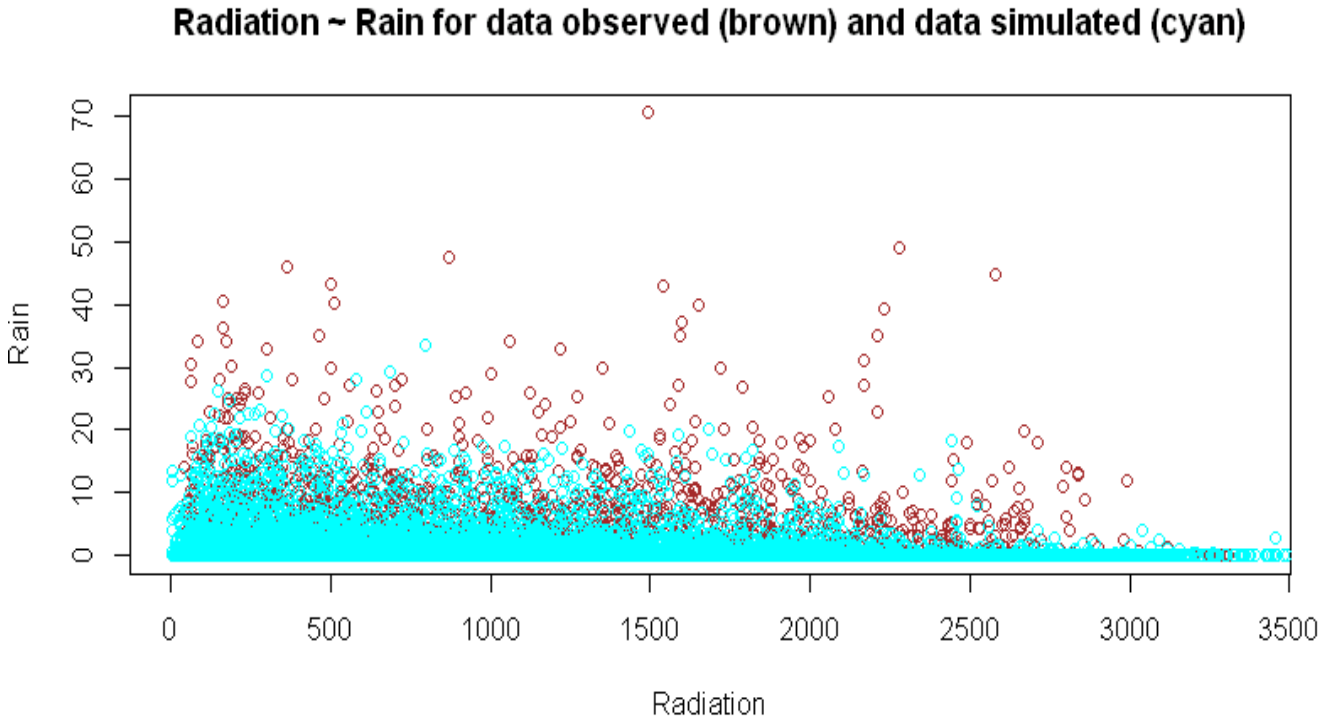

correlation between Tmin Wind on data observed
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9
10
12

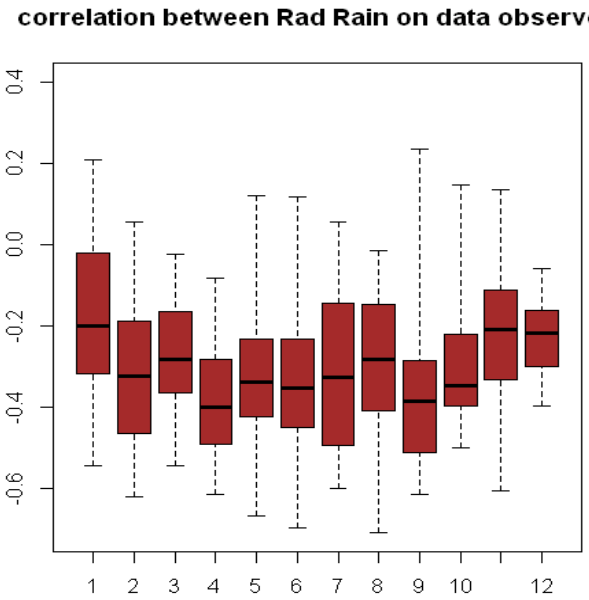

correlation between Tmin Wind on data simulated
-0.4
-0.2
0.0
0.2
0.4
0.6
0.8
1
2
3
4
5
6
7
8
9
10
12

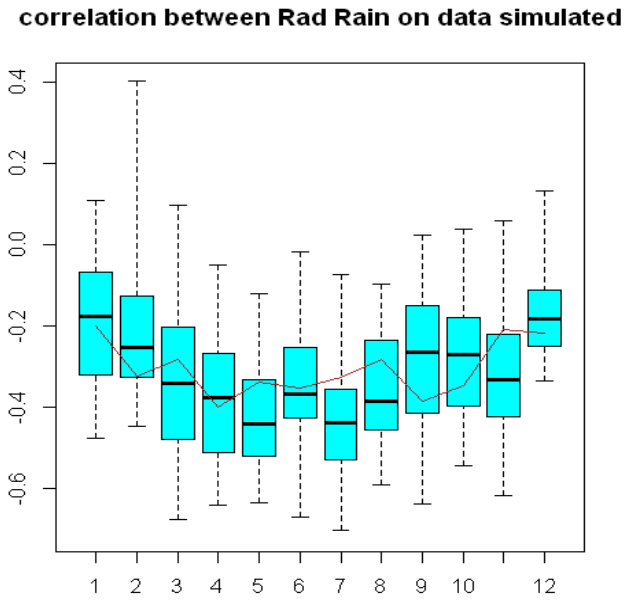

Radiation ~ Tmax for data observed (brown) and data simulated (cyan)
Tmax
-10
0
10
20
30
40
0
500
1000
1500
2000
2500
3000
3500
Radiation


correlation between Rad Tmax on data observed
-0.5
0.0
0.5
1
2
3
4
5
6
7
8
9
10
12


correlation between Rad Tmax on data simulated
-0.5
0.0
0.5
1
2
3
4
5
6
7
8
9
10
12

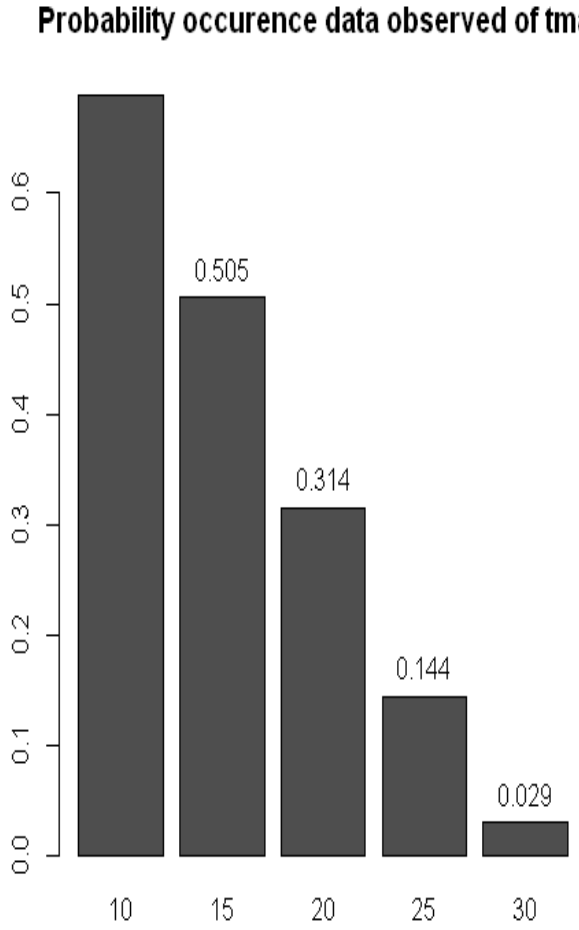

Radiation ~ Rain for data observed (brown) and data simulated (cyan)
Rain
0
10
20
30
40
50
60
70
0
500
1000
1500
2000
2500
3000
3500
Radiation

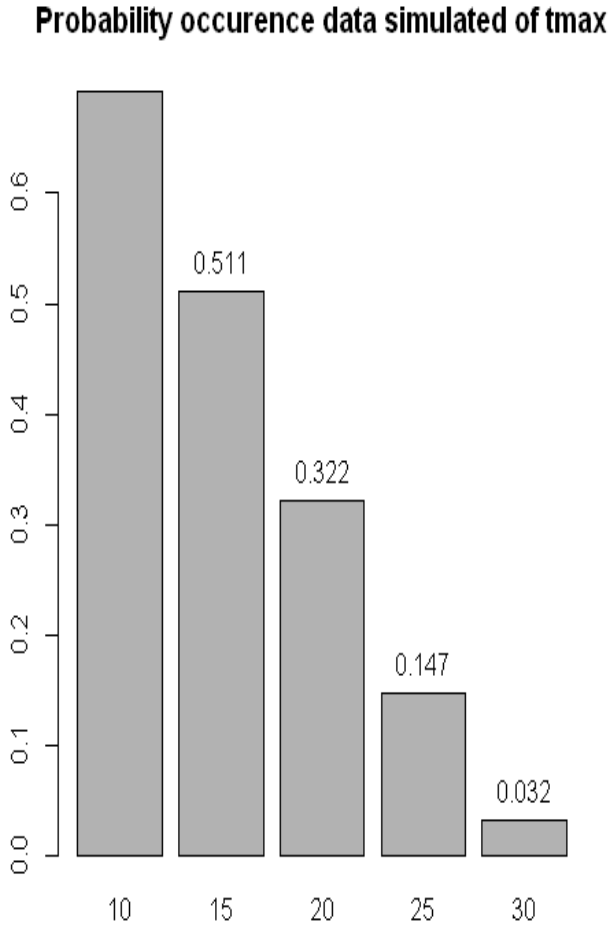

correlation between Rad Rain on data observed
0.4
0.2
0.0
-0.2
-0.4
-0.6
1
2
3
4
5
6
7
8
9
10
12


correlation between Rad Rain on data simulated
0.4
0.2
0.0
-0.2
-0.4
-0.6
1
2
3
4
5
6
7
8
9
10
12

Probabilités d'occurrence d'un événement

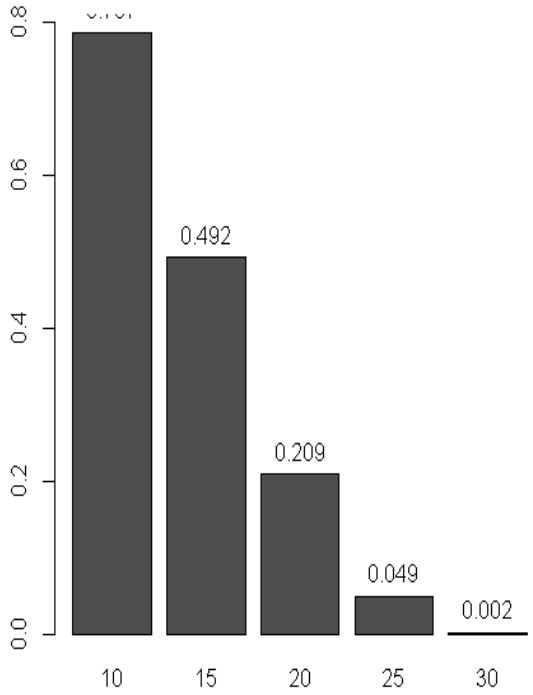
Global :



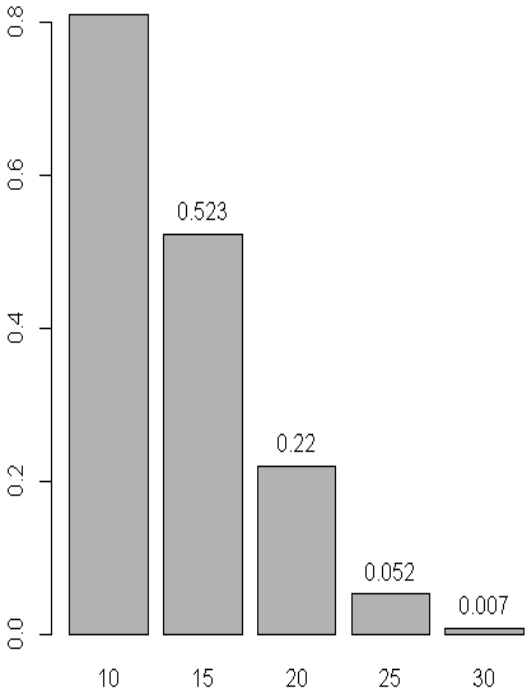


By season (MAM & JJA) :

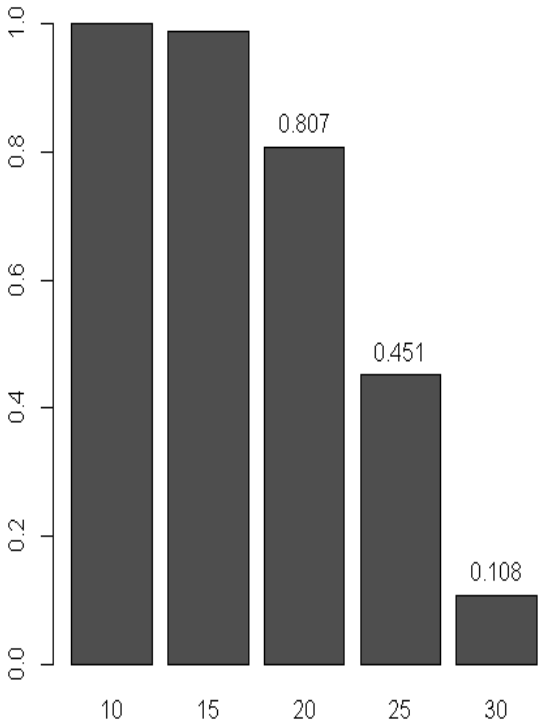
Probability occurence data observed of tmax MAM



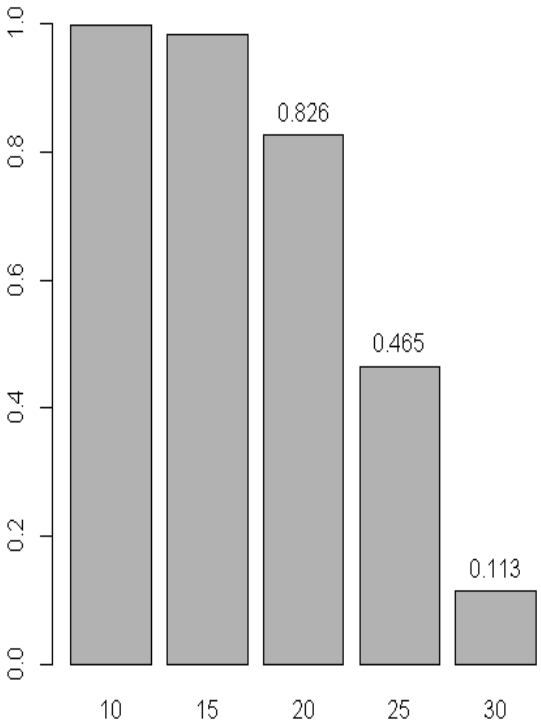
Probability occurence data simulated of tmax MAM



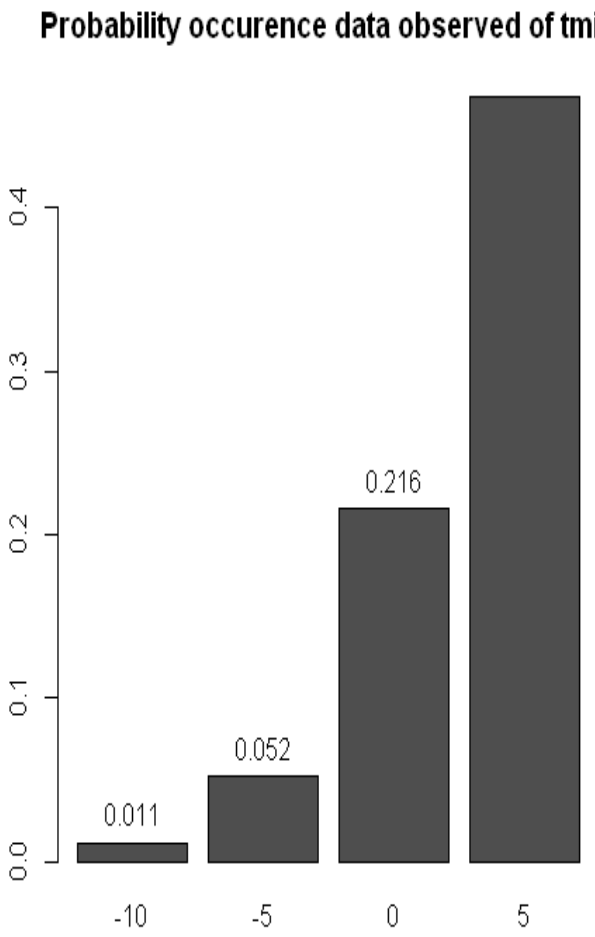
Probability occurence data observed of tmax JJA

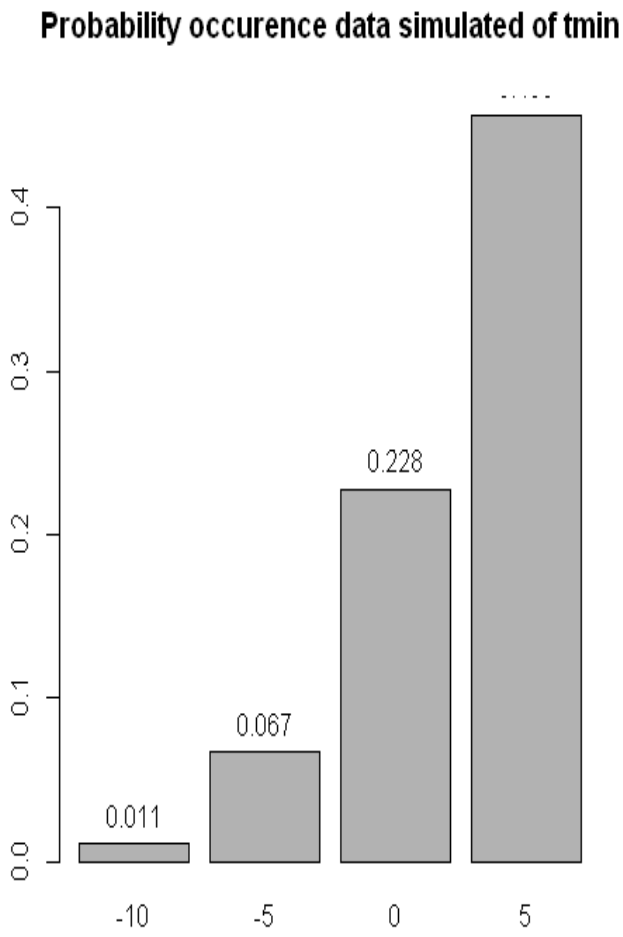


Probability occurence data simulated of tmax JJA

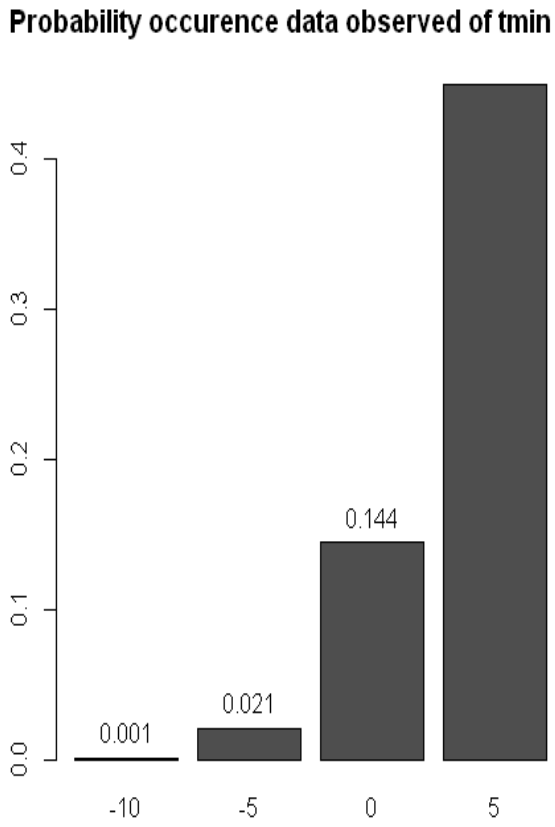


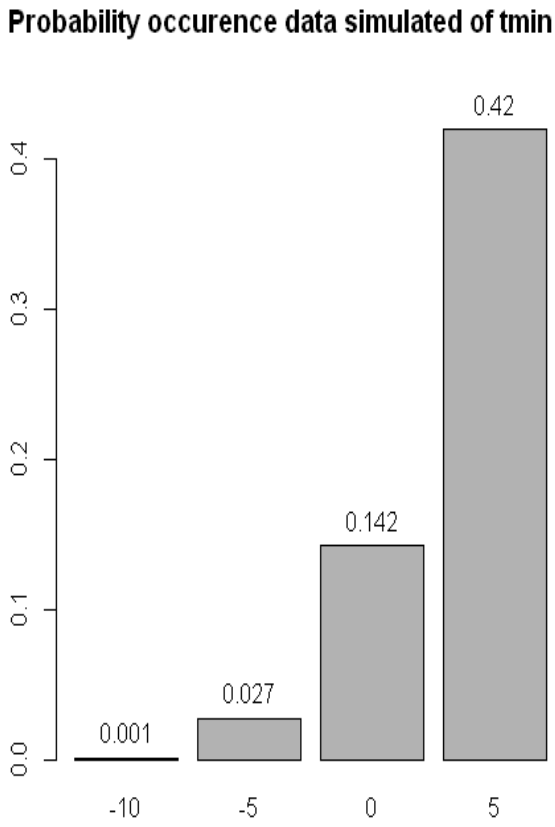
Global :

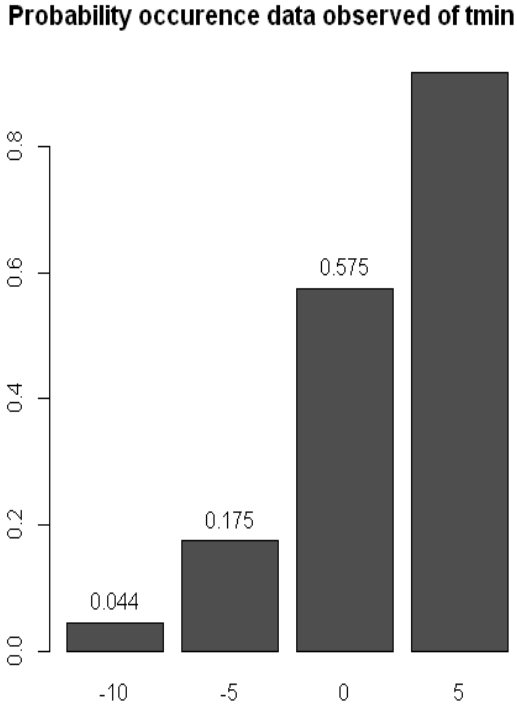


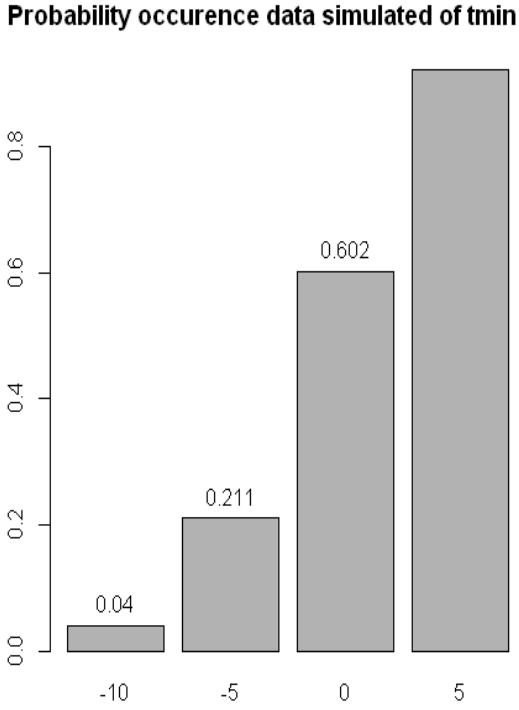


By season (SON et DJF) :


Probability occurence data observed of tmin SON
0.0
0.1
0.2
0.3
0.4
0.001
0.021
0.144
-10
-5
0
5

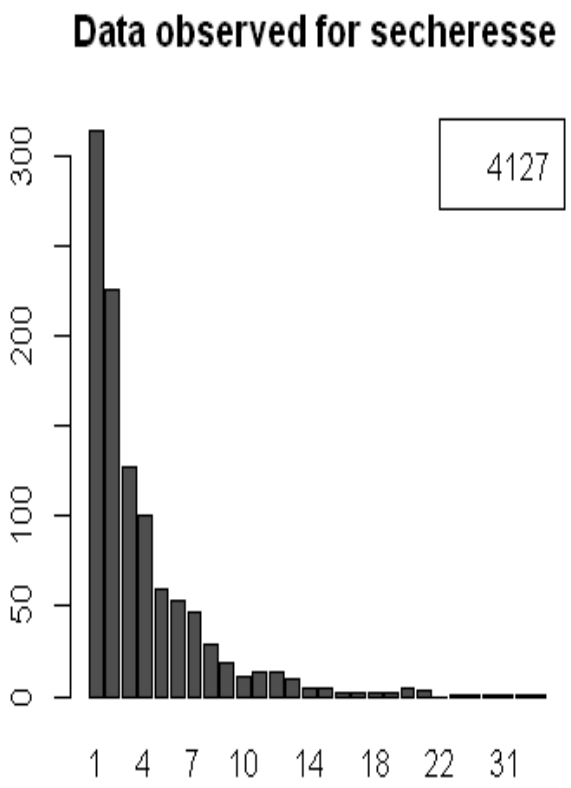

Probability occurence data simulated of tmin SON
0.0
0.1
0.2
0.3
0.4
0.001
0.027
0.142
0.42
-10
-5
0
5

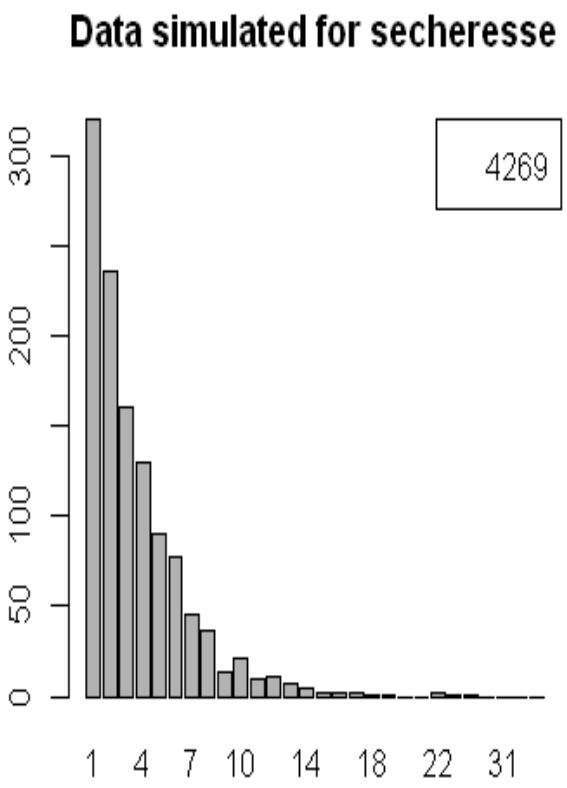

Probability occurence data observed of tmin DJF
0.0
0.2
0.4
0.6
0.8
0.044
0.175
0.575
-10
-5
0
5


Probability occurence data simulated of tmin DJF
0.0
0.2
0.4
0.6
0.8
0.04
0.211
0.602
-10
-5
0
5

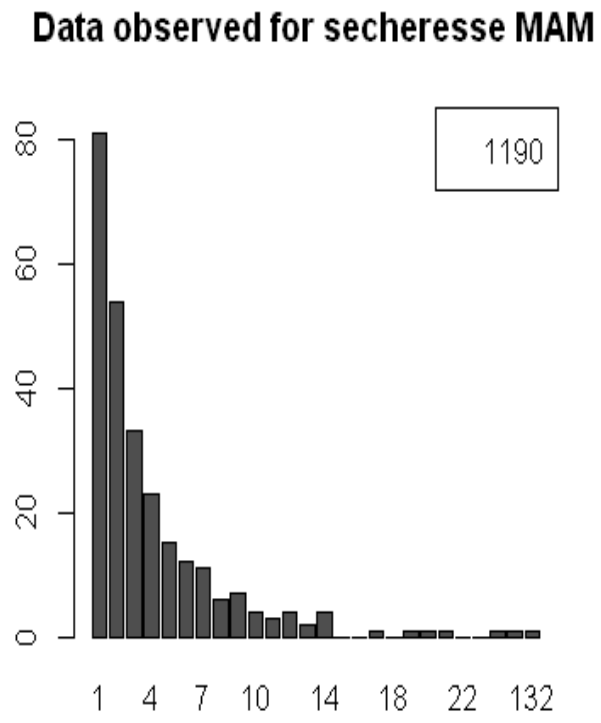
Persistence d'un événement

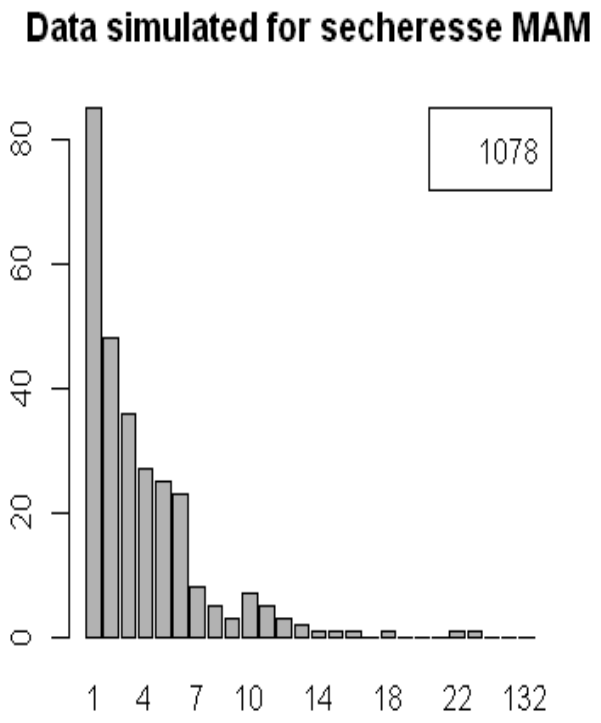
Global :

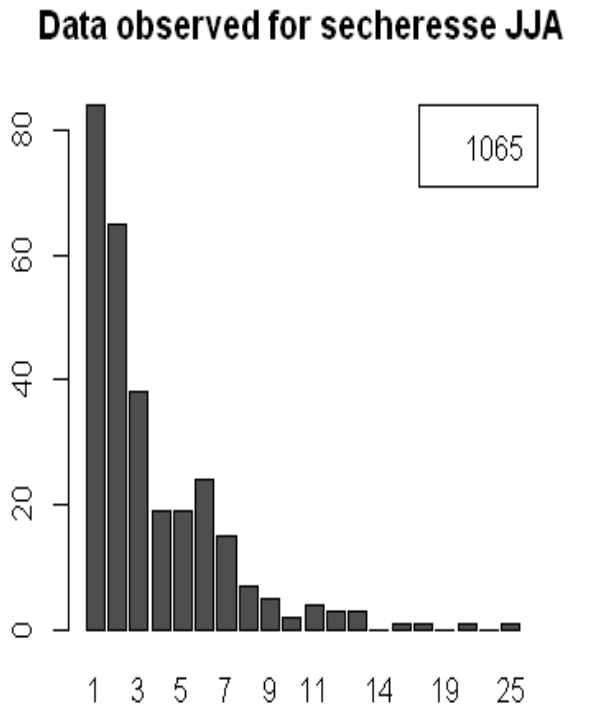


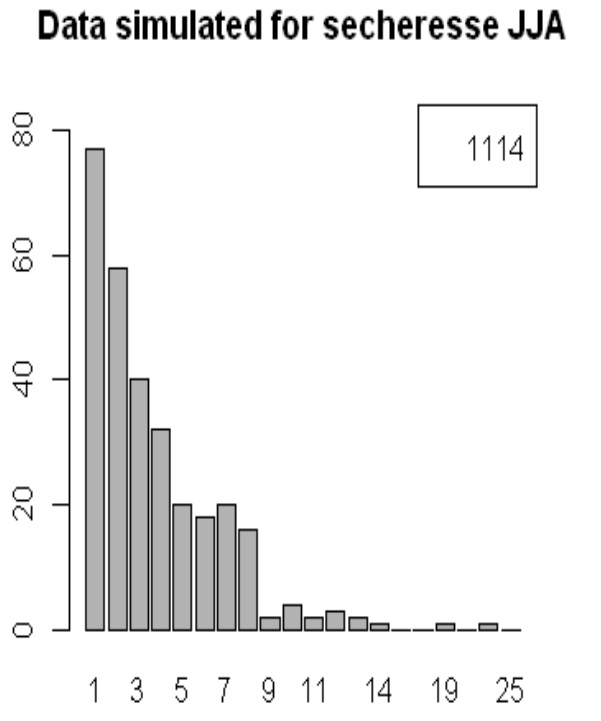


By season (MAM & JJA) :

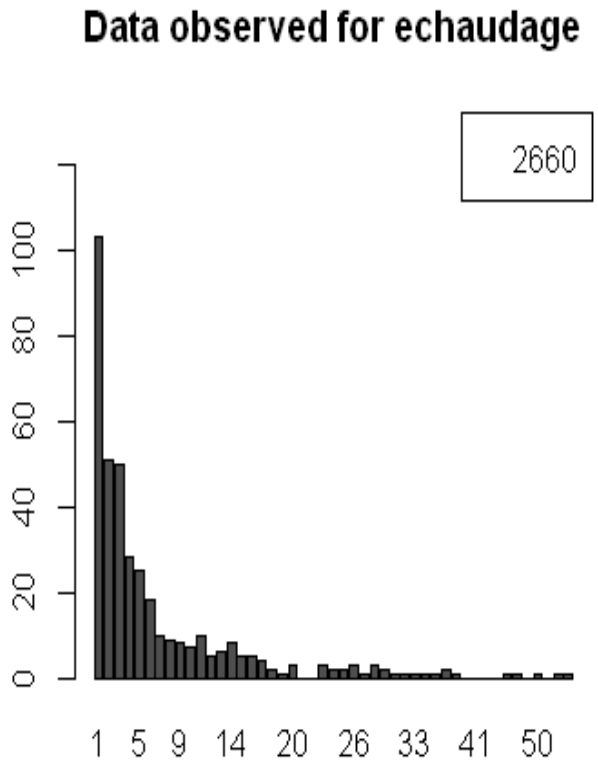

Data observed for secheresse MAM
1190
0
20
40
60
80
1
4
7
10
14
18
22
132

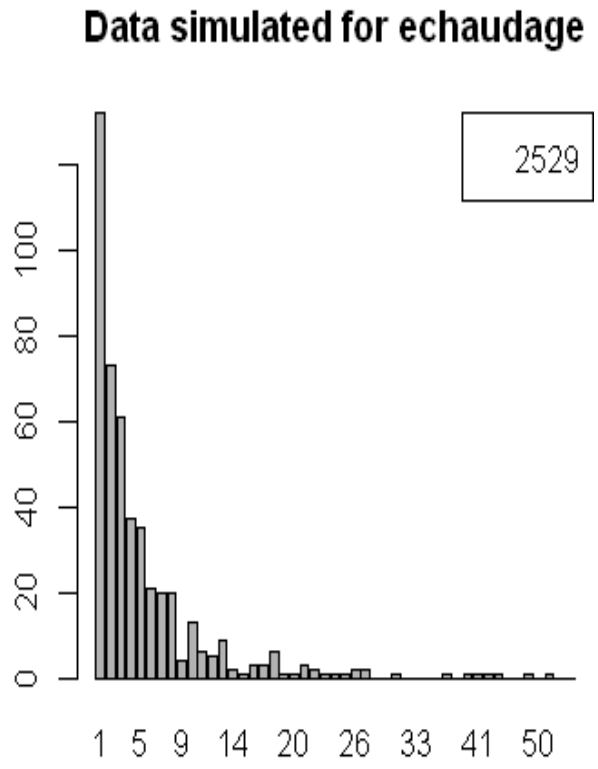

Data simulated for secheresse MAM
1078
0
20
40
60
80
1
4
7
10
14
18
22
132


Data observed for secheresse JJA
1065
0
20
40
60
80
1
3
5
7
9
11
14
19
25

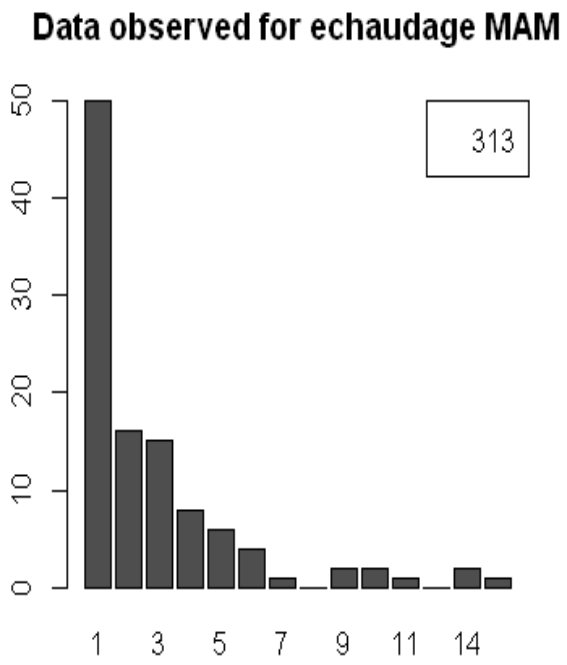

Data simulated for secheresse JJA
1114
0
20
40
60
80
1
3
5
7
9
11
14
19
25

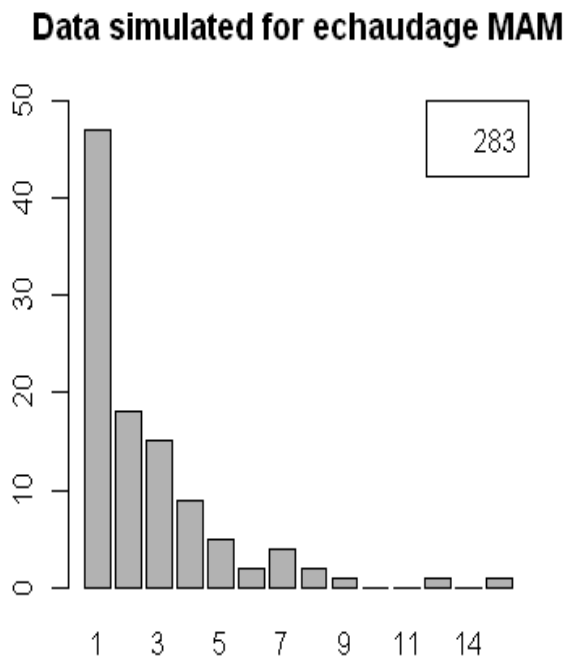
Global :

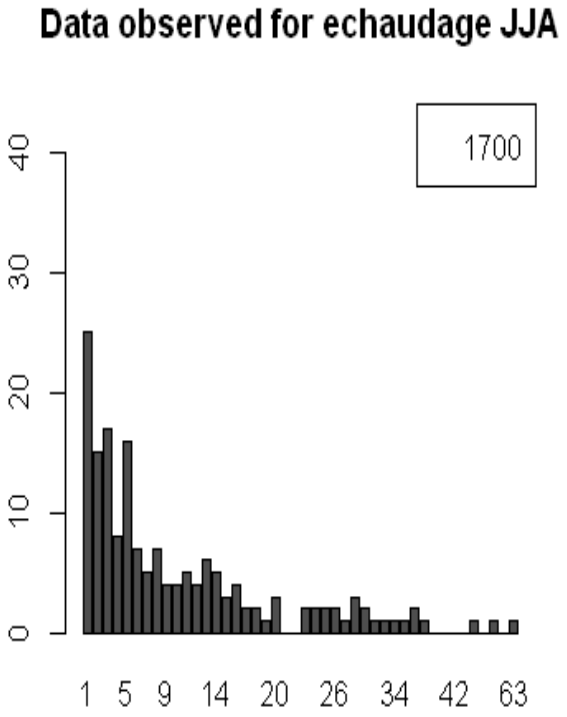


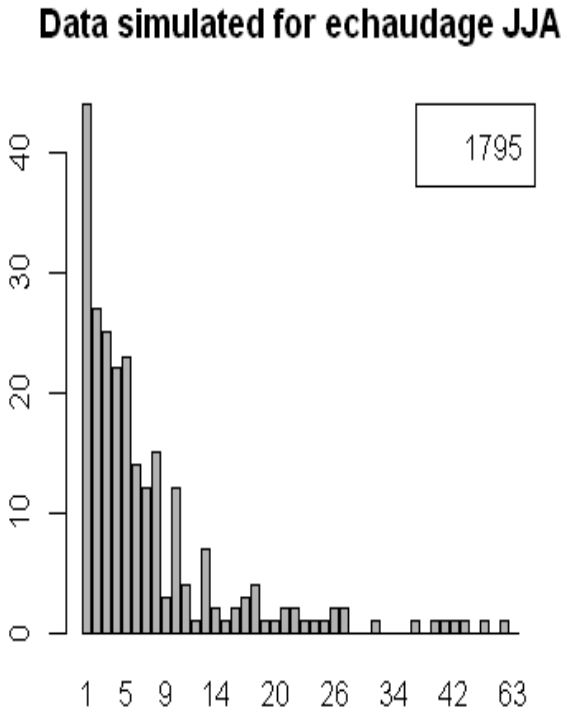


By season (MAM & JJA) :

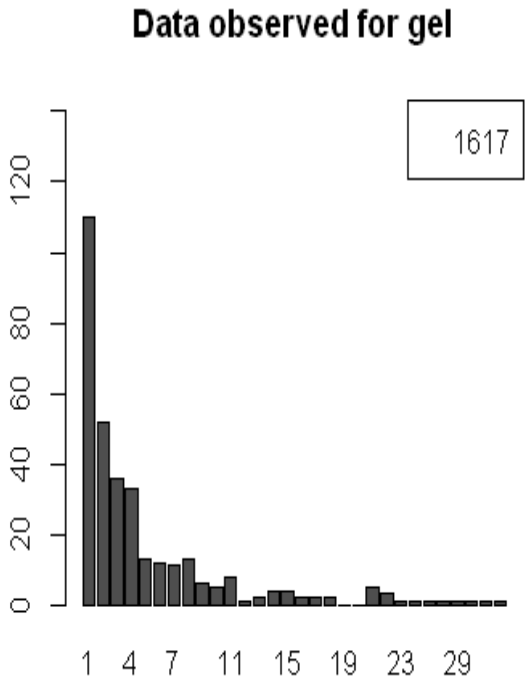

Data observed for echaudage MAM
313
0
10
20
30
40
50
1
3
5
7
9
11
14

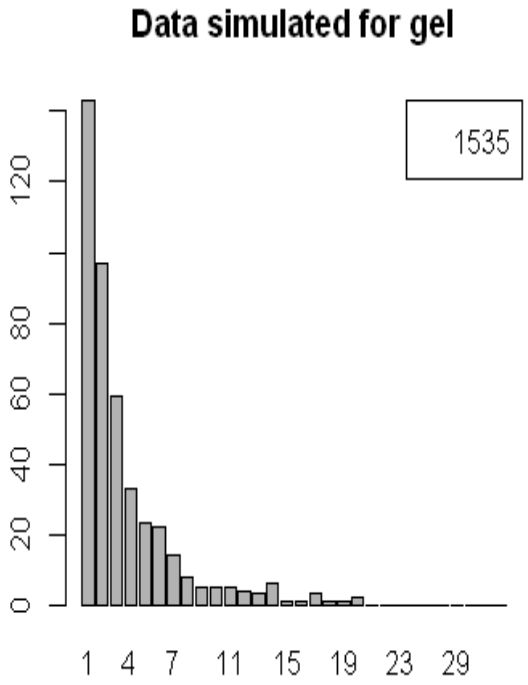

Data simulated for echaudage MAM
283
0
10
20
30
40
50
1
3
5
7
9
11
14


Data observed for echaudage JJA
1700
0
10
20
30
40
1
5
9
14
20
26
34
42
63

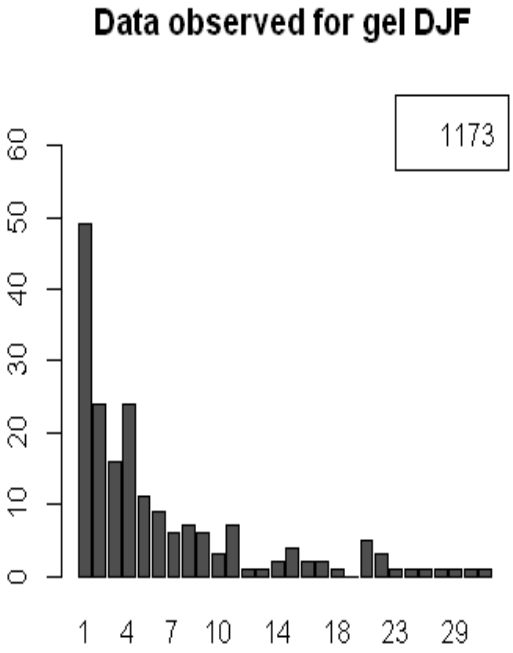

Data simulated for echaudage JJA
1795
0
10
20
30
40
1
5
9
14
20
26
34
42
63

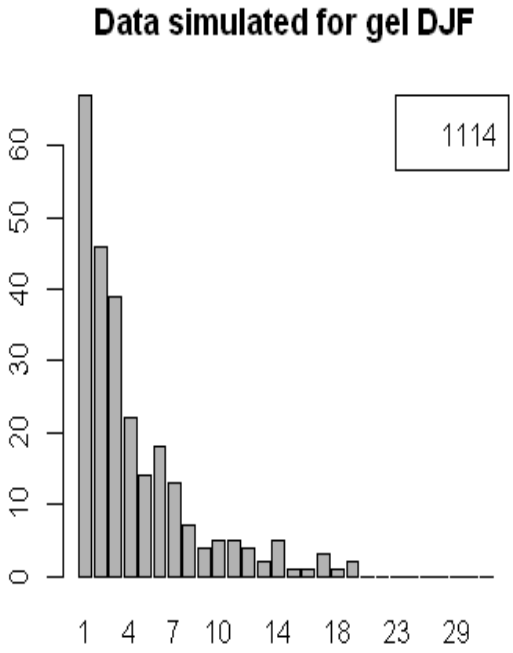
Global :





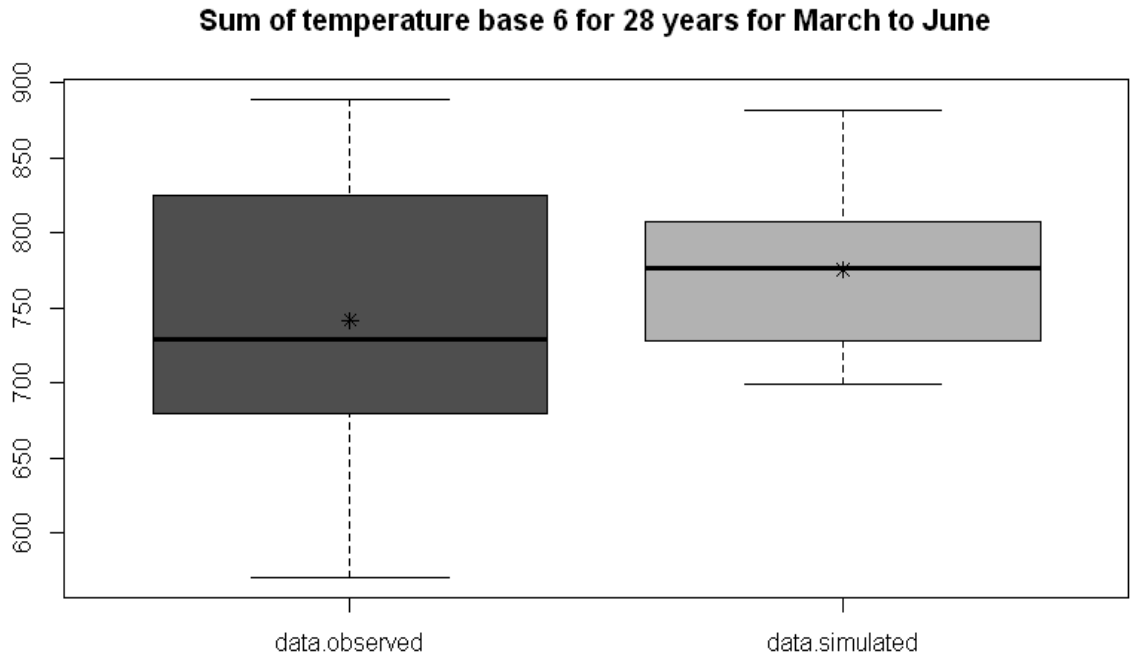
By season (DJF) :





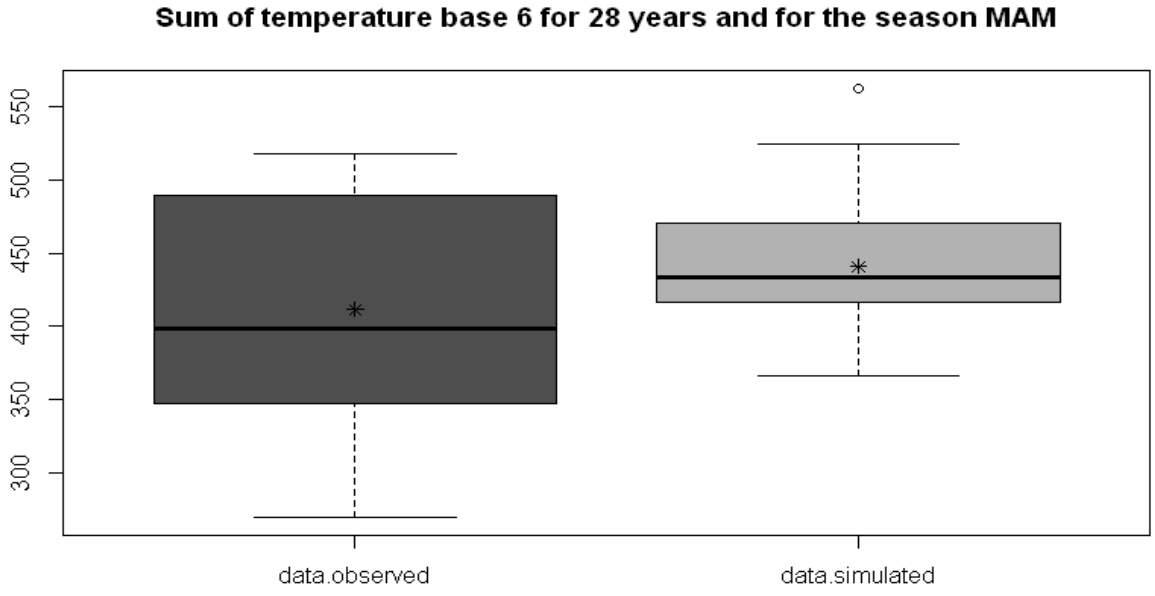
Somme température en base 6

Cycle (March to June) :



[1] "For 28 years, the sum of temperature in base 6 for March to June is 20770.6 for data observed and 21714.95 for data simulated. For one year, average sum for this cycle is 741.81 for data observed and 775.53 for data simulated"

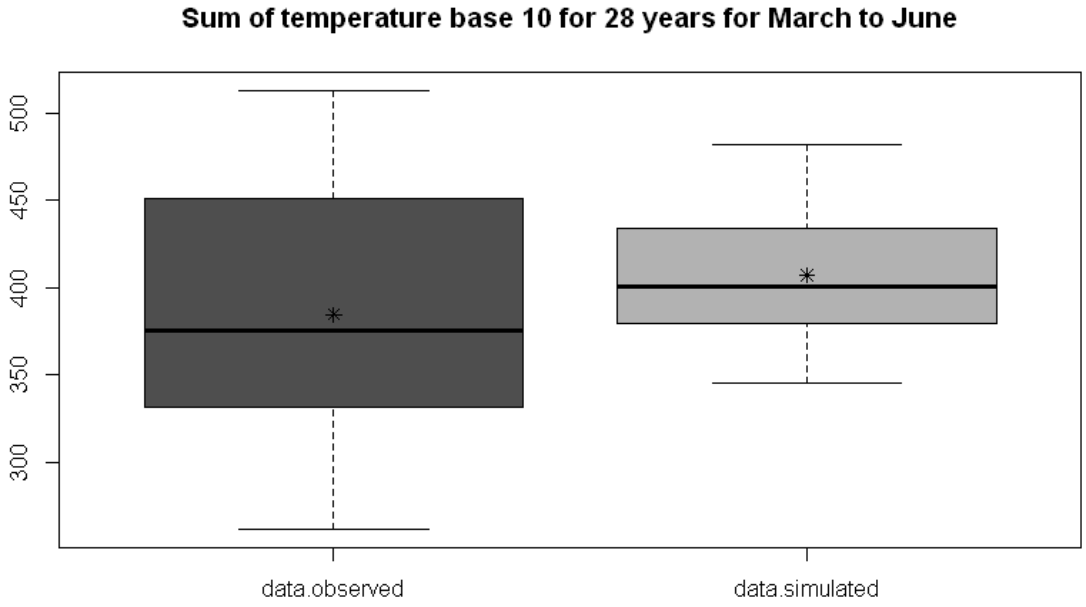
Season (MMA) :



[1] "For 28 years, the sum of temperature in base 6 for the season MAM is 11515.9 for data observed and 12345.45 for data simulated. For one year, average sum for this season is 411.28 for data observed and 440.91 for data simulated"

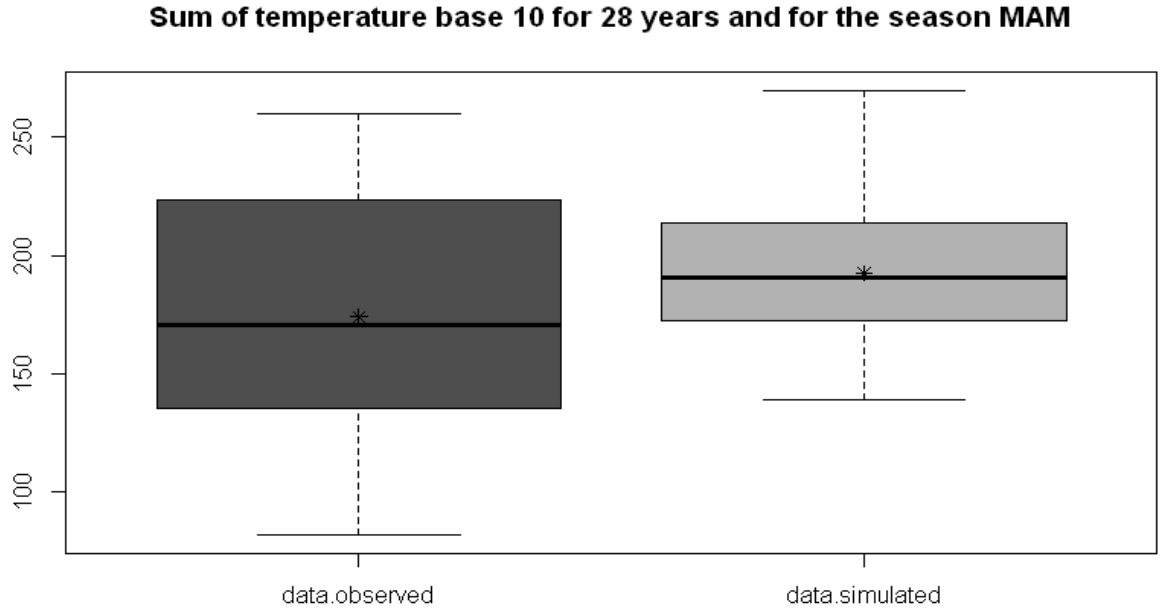
Somme température en base 10

Cycle (March to June) :



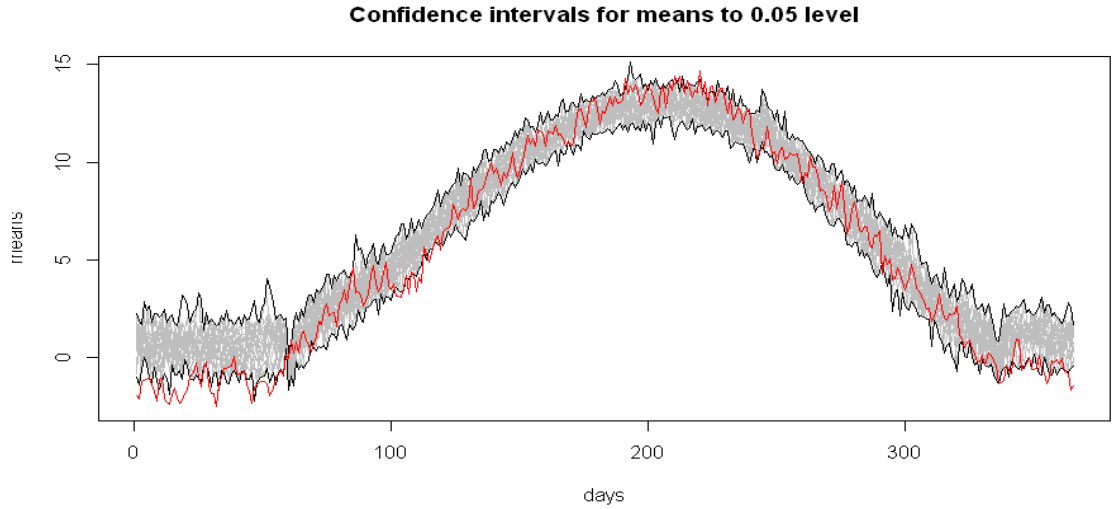
[1] "For 28 years, the sum of temperature in base 10 for March to June is 10771.2 for data observed and 11403.6 for data simulated. For one year, average sum for this cycle is 384.69 for data observed and 407.27 for data simulated"

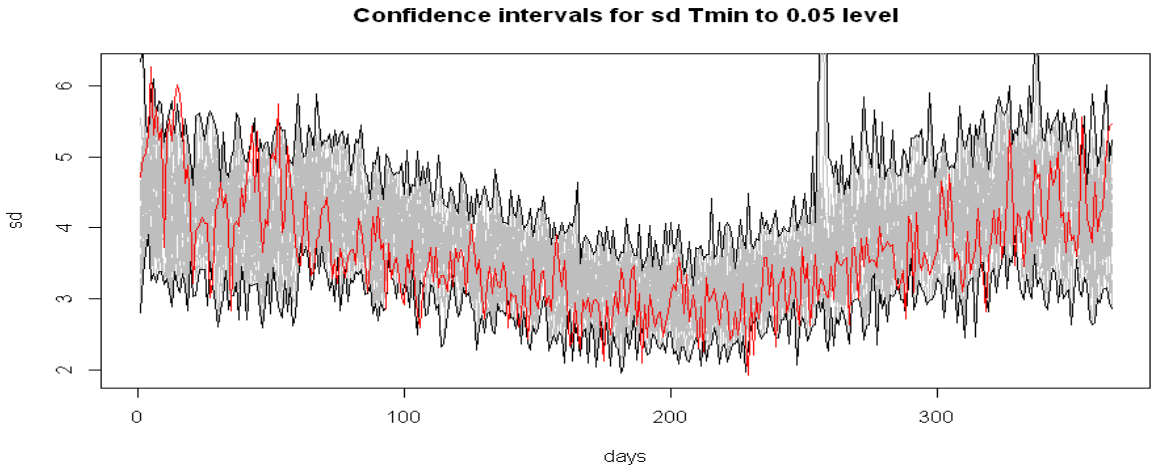
Season (MMA) :

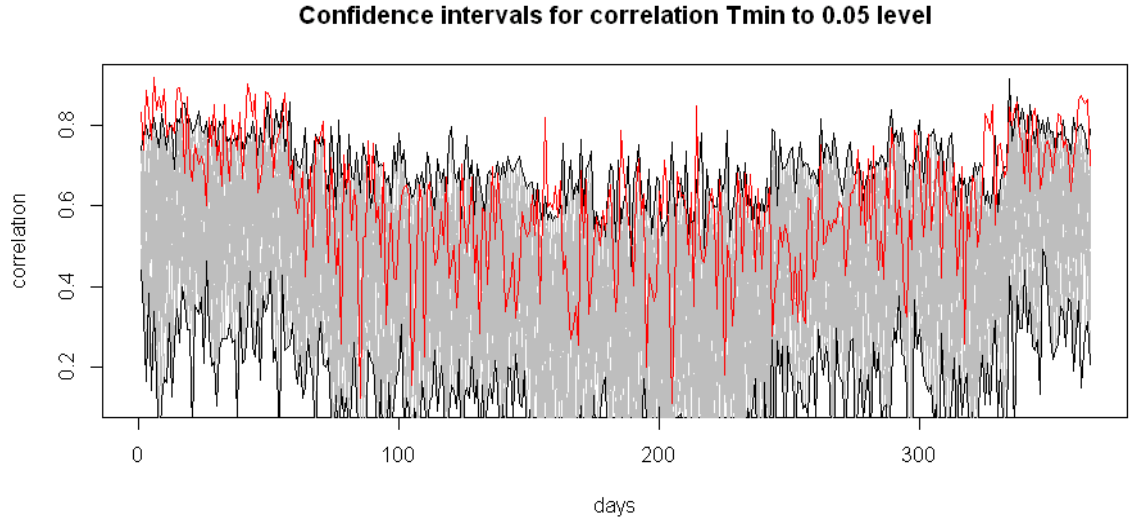


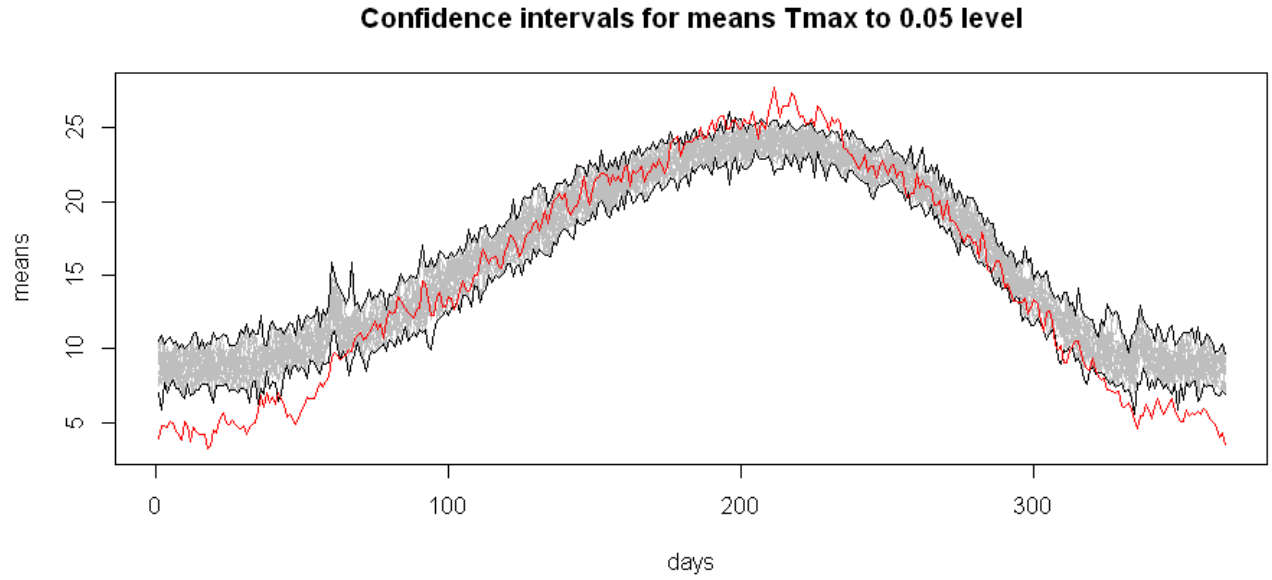
[1] "For 28 years, the sum of temperature in base 10 for the season MAM is 4875.95 for data observed and 5387.4 for data simulated. For one year, average sum for this season is 174.14 for data observed and 192.41 for data simulated"

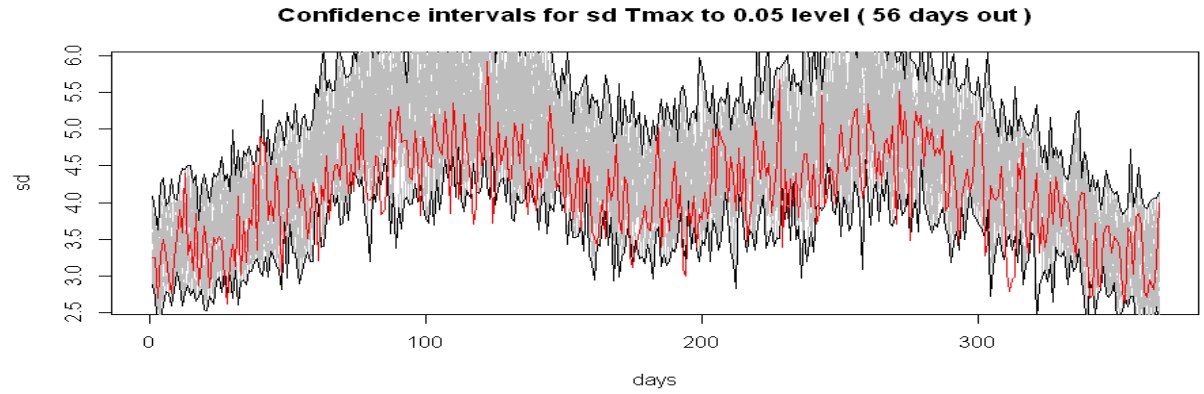
Enveloppes de confiance (alpha = 0,05)

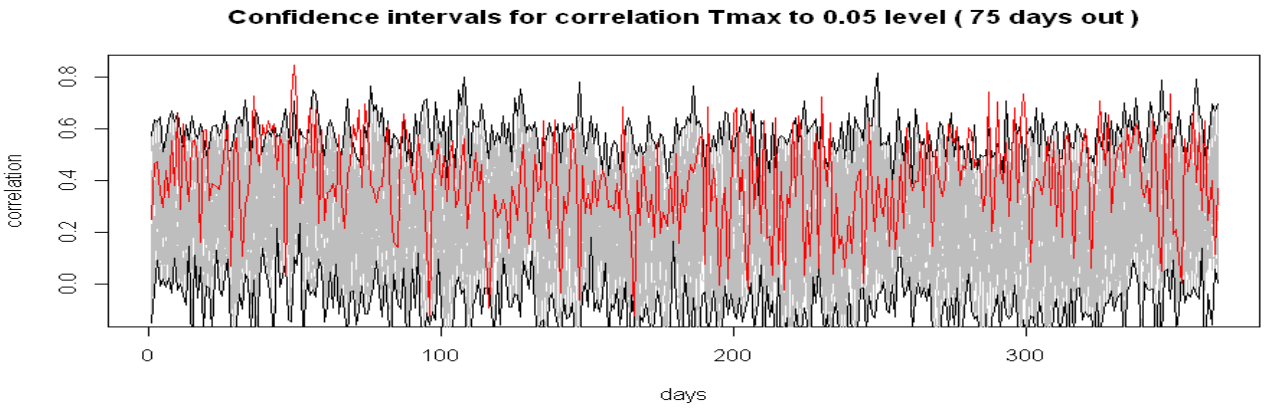


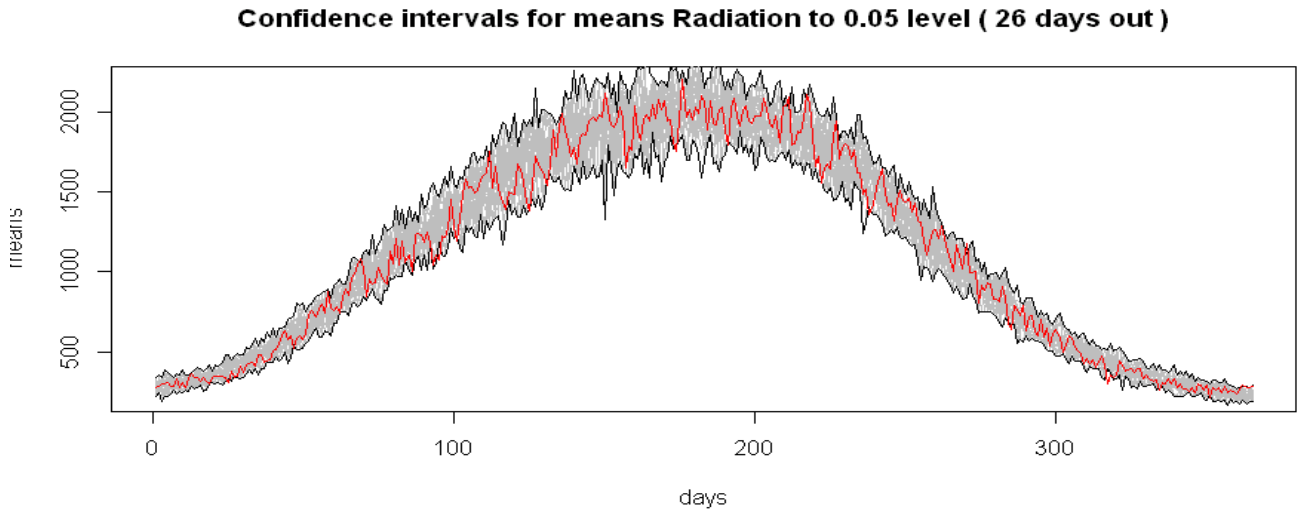


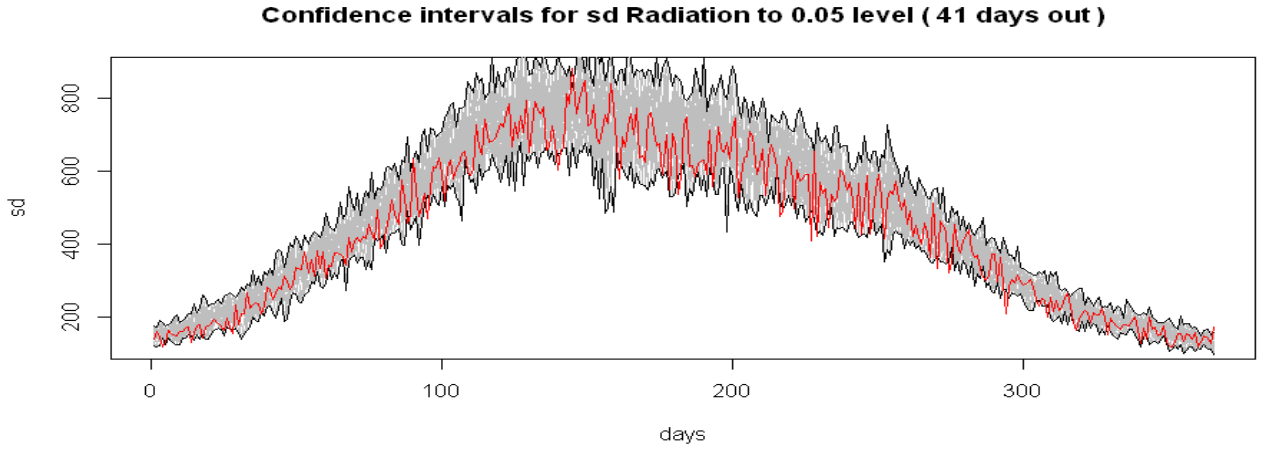


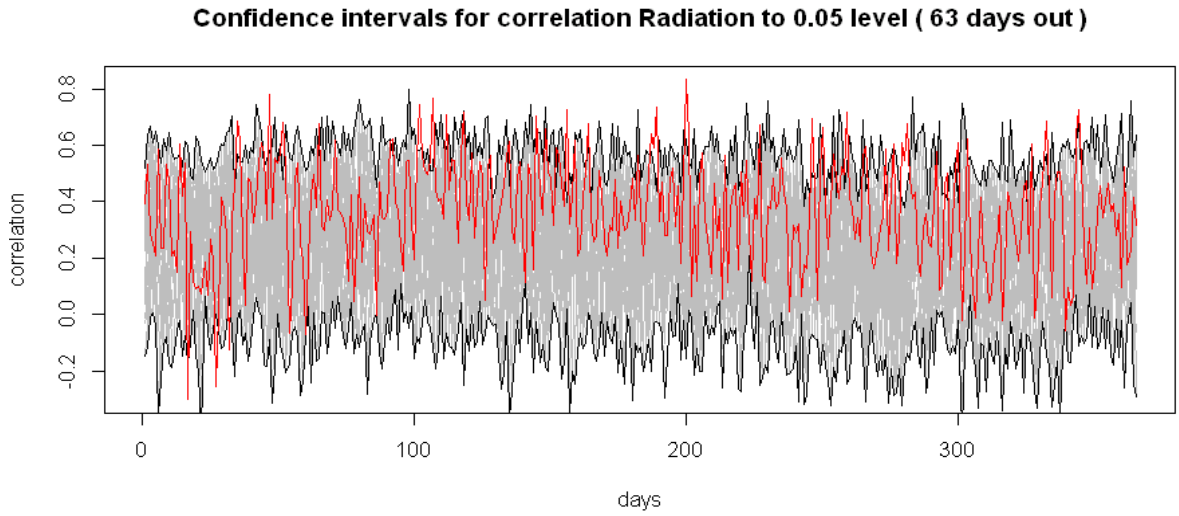

Confidence intervals for means Tmax to 0.05 level
means
5
10
15
20
25
0
100
200
300
days

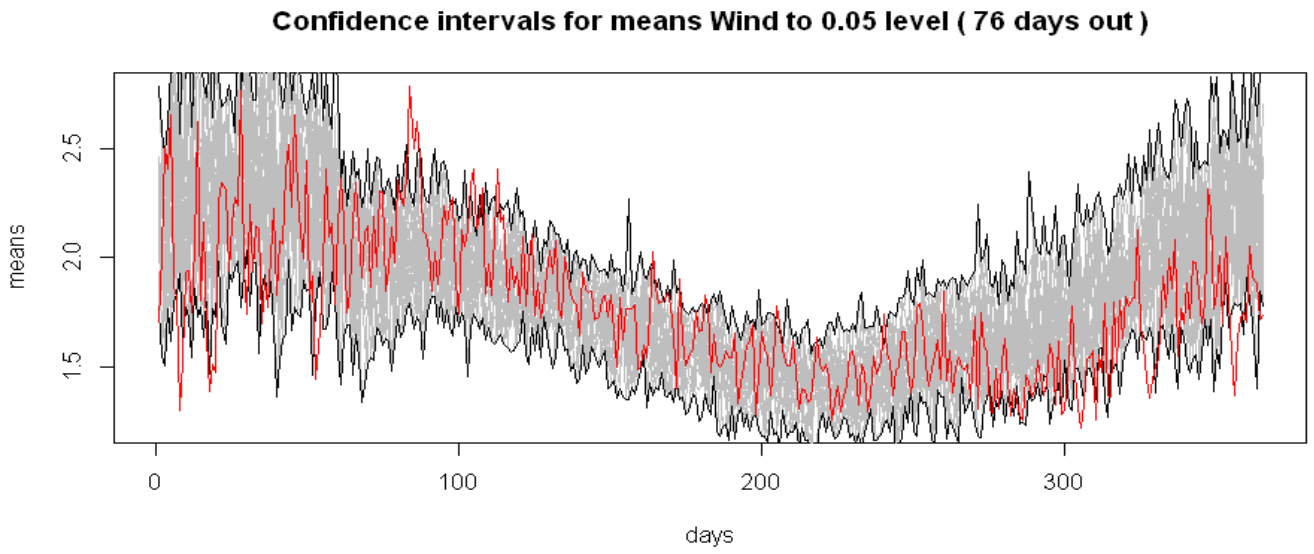

Confidence intervals for sd Tmax to 0.05 level (56 days out)
sd
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
0
100
200
300
days

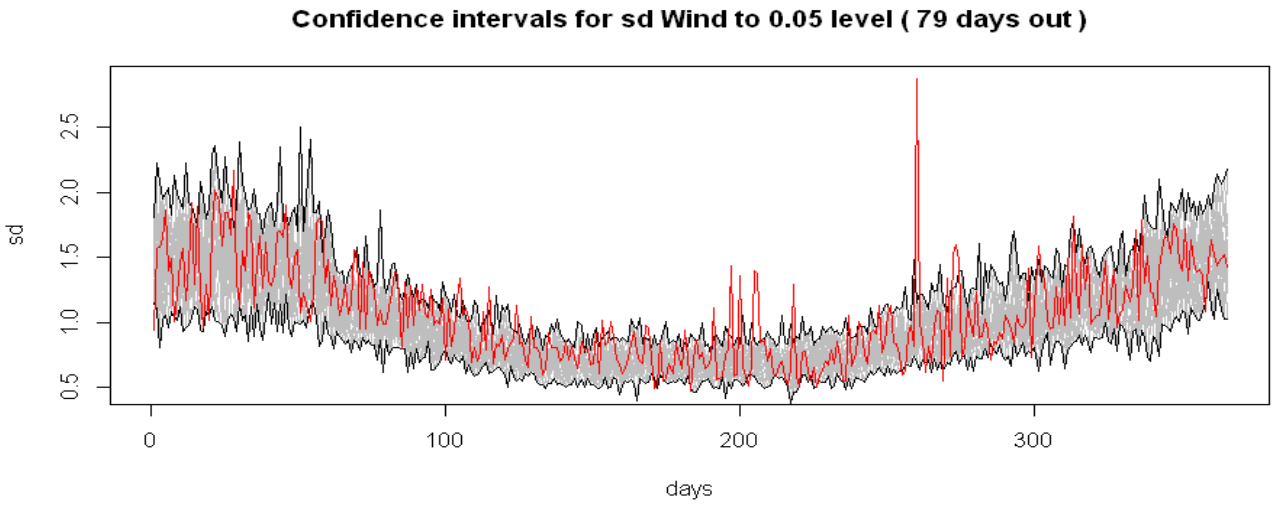

Confidence intervals for correlation Tmax to 0.05 level (75 days out)
correlation
0.0
0.2
0.4
0.6
0.8
0
100
200
300
days

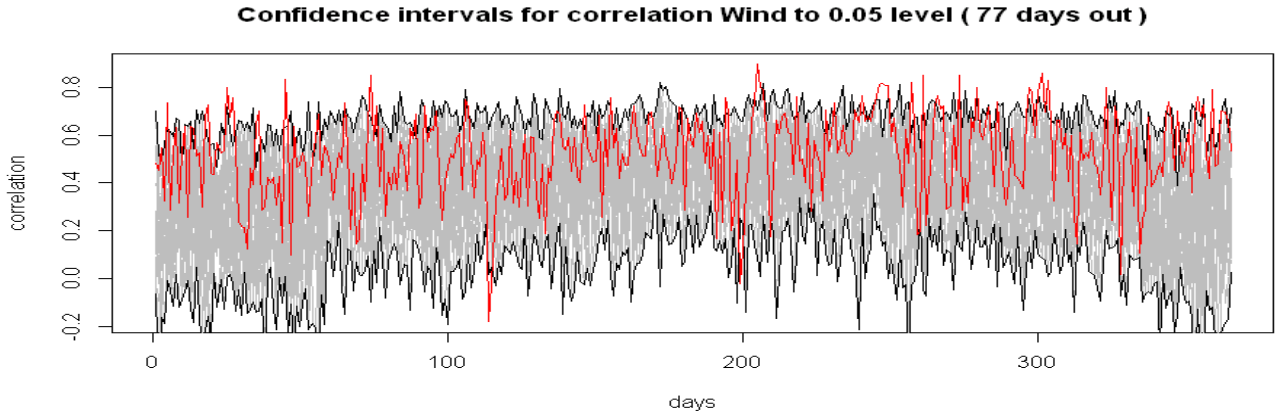

Confidence intervals for means Radiation to 0.05 level (26 days out)
means
500
1000
1500
2000
0
100
200
300
days


Confidence intervals for sd Radiation to 0.05 level (41 days out)
sd
200
400
600
800
0
100
200
300
days


Confidence intervals for correlation Radiation to 0.05 level (63 days out)
correlation
-0.2
0.0
0.2
0.4
0.6
0.8
0
100
200
300
days


Confidence intervals for means Wind to 0.05 level (76 days out)
means
1.5
2.0
2.5
0
100
200
300
days


Confidence intervals for sd Wind to 0.05 level (79 days out)
sd
0.5
1.0
1.5
2.0
2.5
0
100
200
300
days


Confidence intervals for correlation Wind to 0.05 level (77 days out)
correlation
-0.2
0.0
0.2
0.4
0.6
0.8
0
100
200
300
days