Universiteit Hasselt

Faculteit Wetenschappen

SMOOTHERS REGRESSIE

Lineaire Statistische Modellen

Tuur Willio

Contents

1 Introductie

We introduceren de theorie met een probleem. We beschouwen de *Tasmanian Muttonbird* of de Dunbekpijlstormvogel. Deze vogel heeft een speciale opvoedingsmethode waarbij het zijn kuikens voor lange perioden verlaat en dan terug komt en zijn kuikens rijkelijk voedt.
We hebben data van 2 kuikens hun gewicht tot ze het nest verlaten, en niet meer kuikens zijn. Als we een model kunnen vinden van de toename en afname van het gewicht, kunnen we een goed beeld vormen van deze vogel zijn opvoedingstechnieken. Met andere woorden, zijn terugkomst patroon.

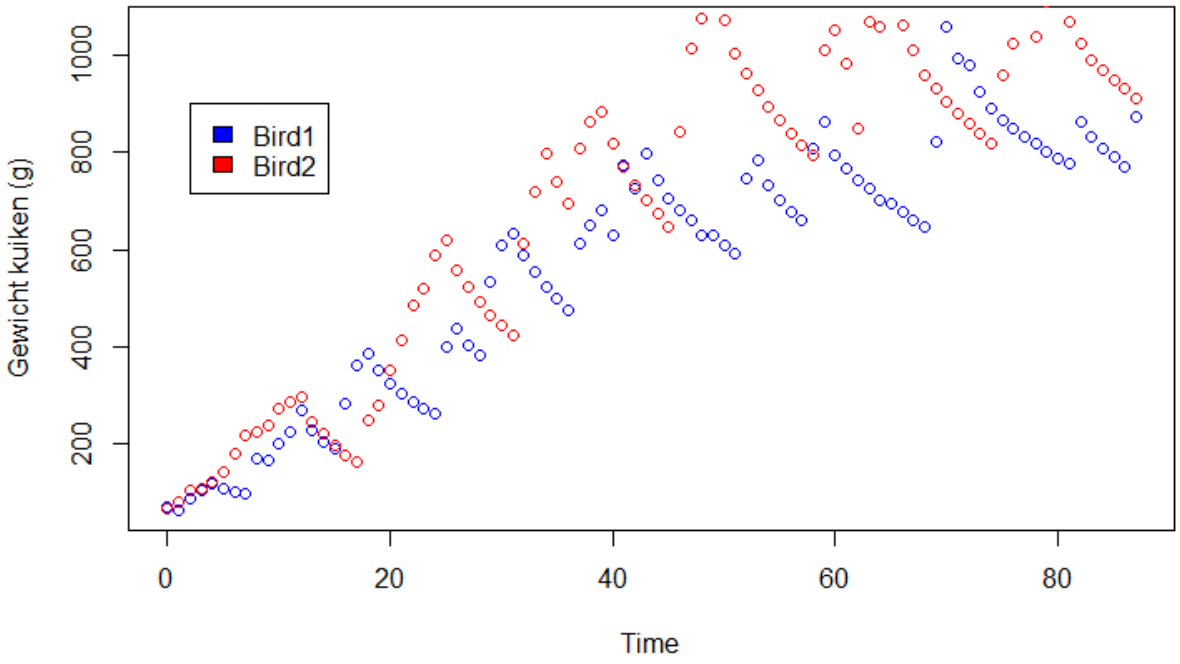


Figure 1: Gegeven Data over gewicht van 2 kuikens
Fisher Island, Bass Strait, 1954

2 Lineaire Regressie

We zullen eerst proberen een lineair model te maken voor de data en de model veronderstellingen voor lineaire regressie na te gaan.
We krijgen een model

$$m(x) = 9.4569x + 158.1269$$

Dit model ziet er niet meteen super slecht uit, maar heeft precies wel wat

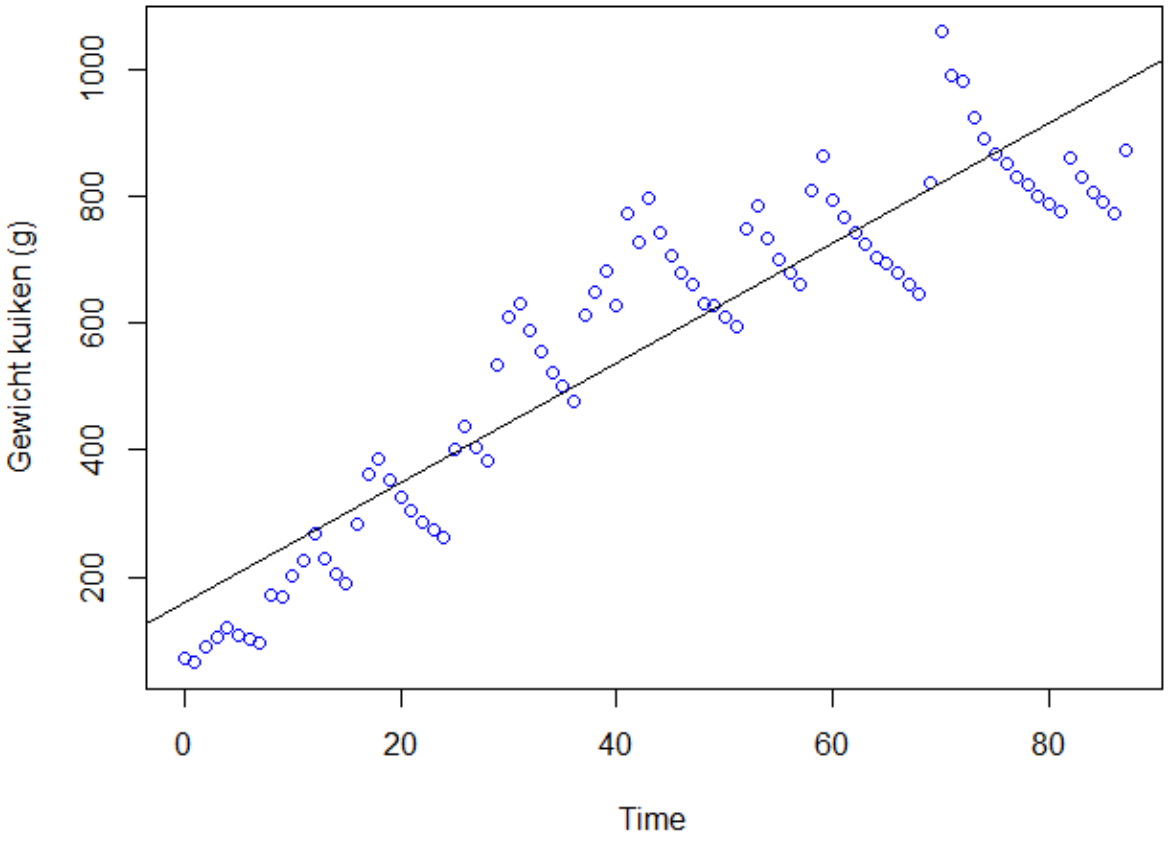


Figure 2: Lineaire model van data

grote residuen. We zullen deze bekijken en zo de lineariteit van het model na gaan.

```
> m = lm(Bird1~Time, data=dataa)
> res=resid(m)
> mean(res)
[1] 5.402639e-15
```

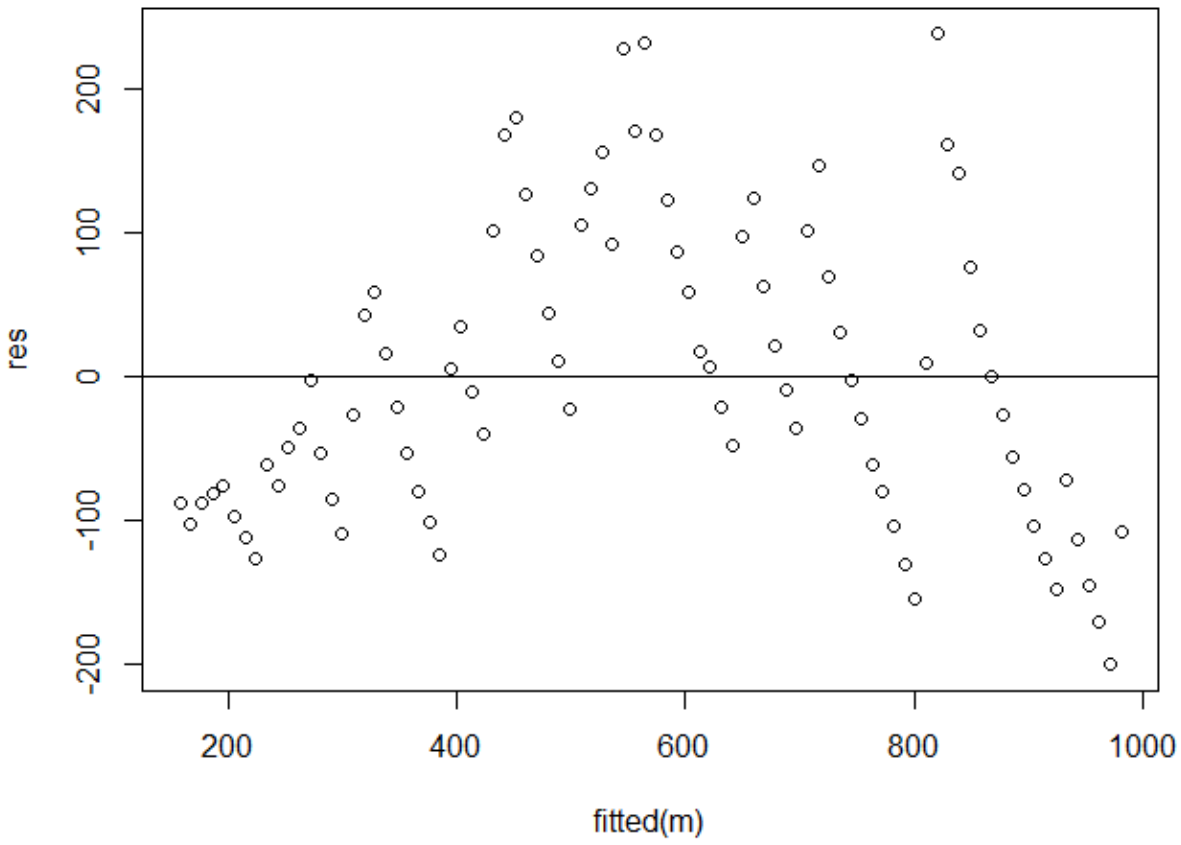


Figure 3: Residuen van het model

We zien dat het gemiddelde van de residuen zeer klein is dus zo goed als 0 is. Dit is een gewenst resultaat, maar als we de residuen plotten zien we dat deze niet zo willekeurig zijn en er precies wel een patroon in voorkomt, niet gewenst voor een lineair model
Om zeker te zijn van ons stuk zullen we ook de kwantielen van de residuen eens plotten over een normale verdeling en zo de normaliteit van de foutterm bestuderen. De QQ-plot toont een duidelijke afwijking bij negatieve residuen.

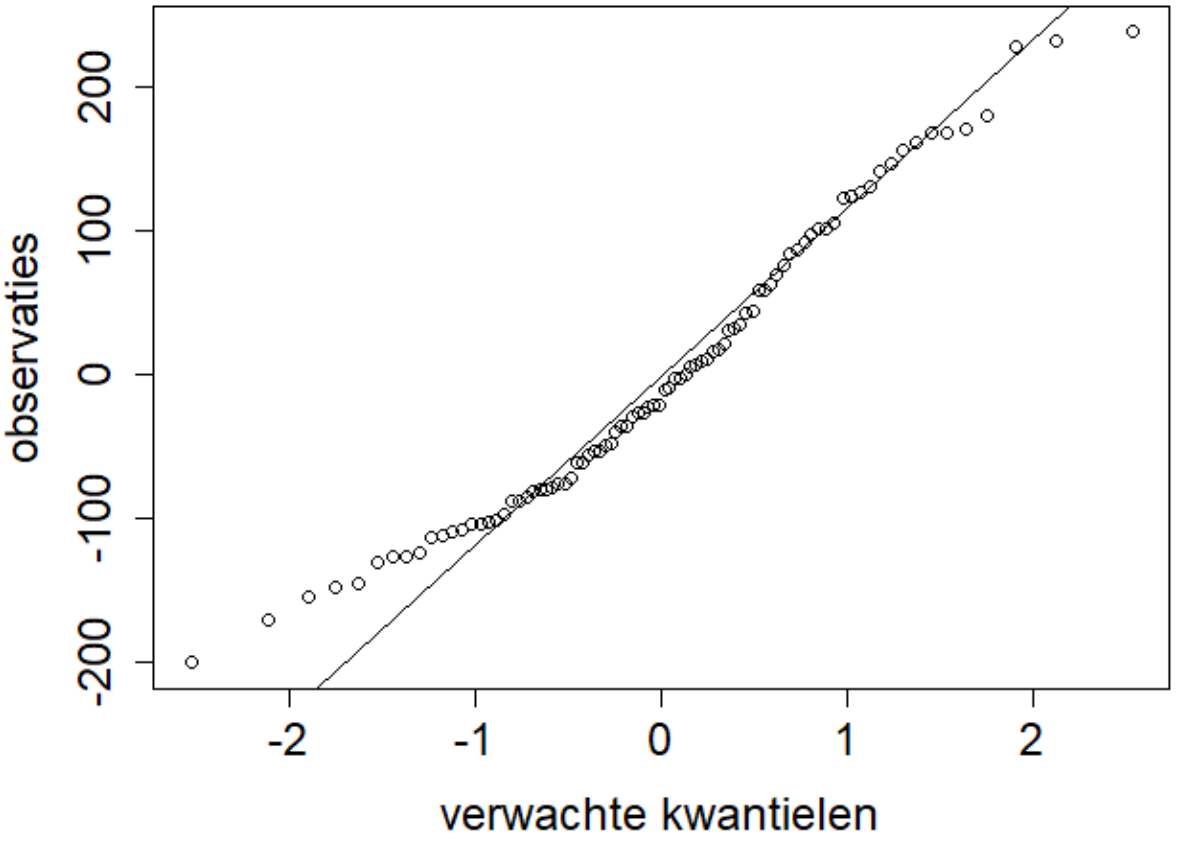


Figure 4: QQ-plot van het model

We besluiten dus dat een lineair model voor deze data niet goed genoeg is.

3 Nonparametric Regression

Niet-parametrische regressie is een deel van de regressie-analyse waarbij we niets afweten van de relatie tussen de regressor en de respons en er geen duidelijk verband zichtbaar is, zoals bijvoorbeeld een lineair verband. Dit verschilt met parametrische regressie waarbij we weten wat het verband is, een bepaalde functie, en waar we dus de parameters van gaan kunnen schatten. Simpele lineaire regressie is een voorbeeld van parametrische regressie waar bij we de vector $\beta^T = (\beta_0, \beta_1)$. met parameters gaan schatten.
Met niet-parametrische regressie zoeken we een functie $m(x)$ die de stijging en daling van Y, de respons, ten opzichte van $X = x$, de regressor goed in kaart brengt zodat:

$$\mathbb{E}[\, Y \mid X = x] = m(x)$$

Smoothers regressie is een onderdeel van niet-parametrische regressie waarbij we vanuit gaan dat $m(x)$ glad is of dus *smooth*

4 Bin Mean

Bij Smooth Regressie gaan we er van uit dat er altijd genoeg data is om een degelijk beeld te geven van het effectieve model, dus niet te weinig op dat het een vertekend beeld geeft. Er zijn veel methode om nu een glad model op te stellen van de data. De meeste vertrekken van hetzelfde principe, opdelen en *'fitted values'* bepalen.
Per regressorwaarde, relevant aan de context en binnen de data, stelt men een partitie op rond deze waarde zoals geïllustreerd in de afbeelding.
Goed om te weten is dat in de afbeelding een cirkel weergeeft die niet juist is met een 1:1 schaal. Het gewicht zit namelijk in een andere grootorde dan de tijd. Dit is makkelijk in orde te brengen door de relatieve fout te gebruiken om de afstand te berekenen.
Zodat we per waarde voor X een groep data punten hebben die ons helpen de geschatte waarde te illusteren.
We kunnen deze bins voor een gegeven x_k opstellen door een interval er rond te nemen zodat we de bin voor x_k kunnen vormen door

$$N(x_k) = \{x \in [x_k - \rho, x_k + \rho] \text{ en } x \in X\}$$

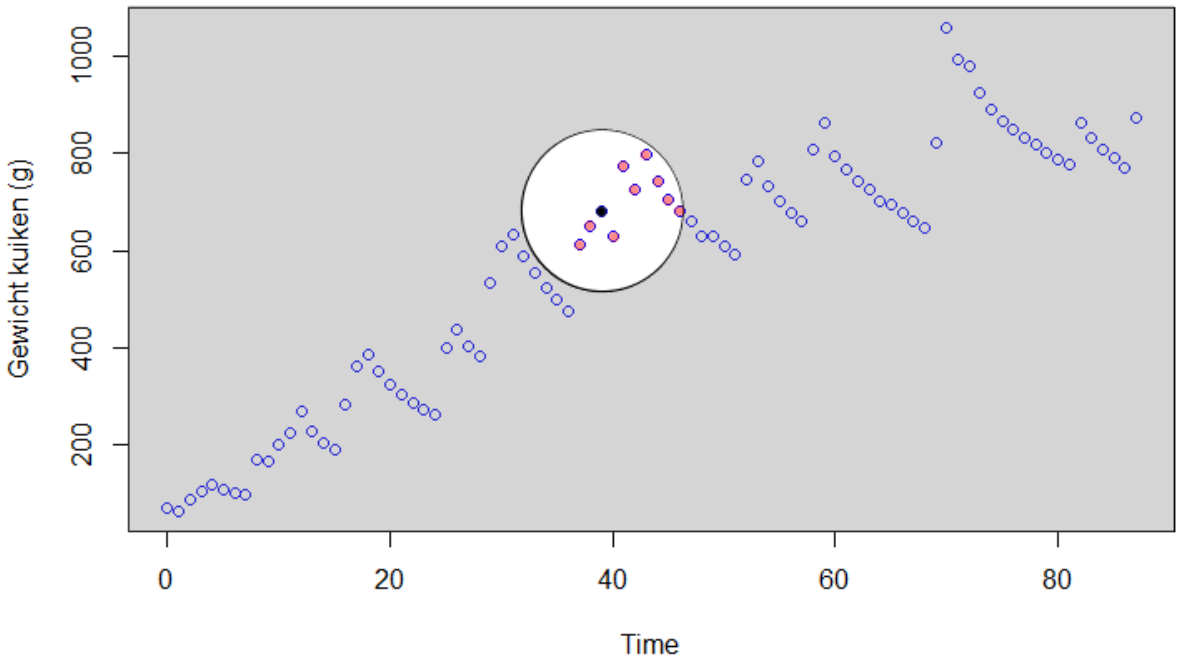


Figure 5: Data met een zichtbare bin

of een andere manier waarbij men een $h = \#datapunten$ neemt en uw intervallen zo te kiezen dat ze de h dichtste datapunten bevatten, en niet meer dan h.

$$N(x_k) = \{x_i \,|\, x_i \ h\text{-}closest\ neighbour\ of\ x_k\}$$

Het principe van bin mean is nu een het model $m(x)$ op te stellen door per x het gemiddelde van de *neighbours* in de bins te nemen.

$$m(x) = \frac{1}{h} \sum_{x_i \in N_h(x)} x_i$$

We kunnen de verwachte waarde van de fout ϵ_i van dit model als volgt noteren, met h de bin grootte.

$$
\begin{aligned}
& \epsilon_i = Y_i - m(x_i) \\
\implies & \mathbb{E}(\epsilon_i) = \mathbb{E}(Y_i) - \mathbb{E}(m(x_i)) \\
\implies & \mathbb{E}(\epsilon_i) = \mu - \frac{1}{N}\sum_{j=1}^{N}(\frac{1}{h}\sum_{x_i \in N_h(x_j)} x_i) \\
\implies & \mathbb{E}(\epsilon_i) = \mu - \frac{1}{h}\sum_{j=1}^{N}\frac{(x_{k1} + ... + x_k + ... + x_{kh})}{N} \\
\implies & \mathbb{E}(\epsilon_i) = \mu - \frac{1}{h}h\mu = 0
\end{aligned}
$$

Dit is dus een goed model!
Met volgende R functie om een *bin* te maken kunnen we Bin mean uitwerken voor de vogels.

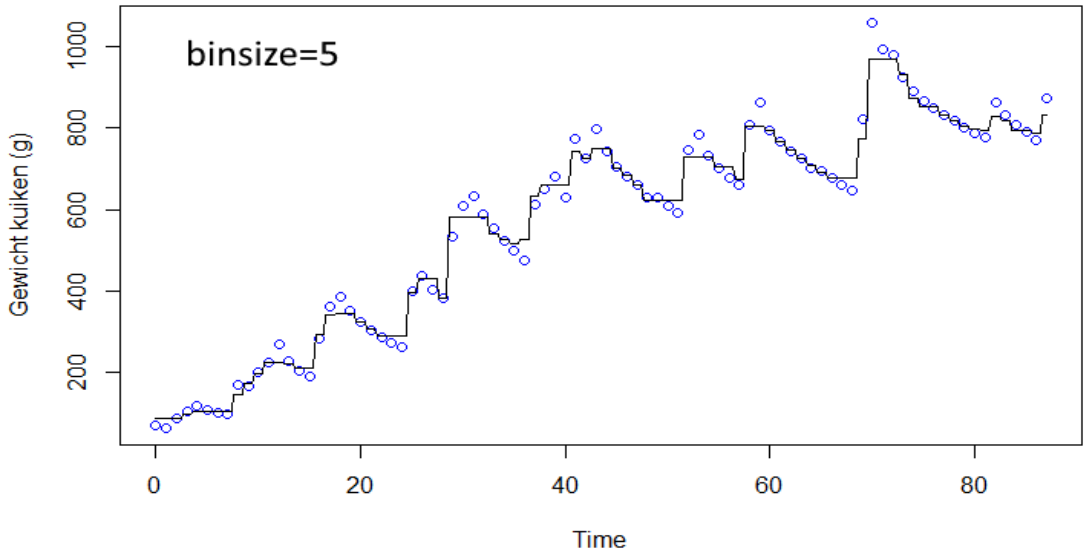
```
bin = function(x,binsize,data=dataa){
  x = round(x)
  x_min = x-binsize
  if(x_min<0){x_min=0}
  x_max = x+binsize
  if(x_max>88){x_max=88}
  #options for bin
  neighbour_opt = data[x_min:x_max,1:2]#BIRD 1 == INDEX 2
  #get relative distance
  for (i in 1:nrow(neighbour_opt)) {
    dis = sqrt( ((neighbour_opt[i,1]-x)/(x+as.integer(x==0)))^2+...
    ...((neighbour_opt[i,2]-data[x+1,2])/data[x+1,2])^2)#BIRD 2
    neighbour_opt[i,3] = dis
  }
  colnames(neighbour_opt)[3]="Distance"
  #get minimum binsize times
  binn= {}
  for(i in 1:binsize){
    min_dis = min(neighbour_opt[,3])
    binn = rbind(binn, neighbour_opt[neighbour_opt[,3] == min_dis,1:2])
```

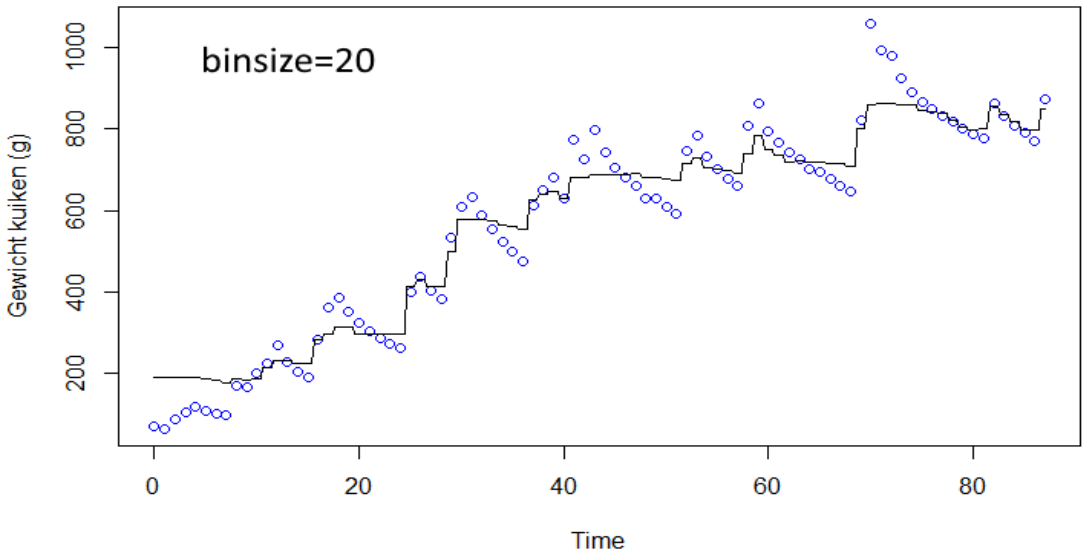
```
    neighbour_opt = subset(neighbour_opt, neighbour_opt[,3]!=min_dis)
  }
  #points(binn[,1],binn[,2],col="red")
  #points(x,data[x+1,2])
  return (binn)
}

bin_smooth = function(x,binsize, dataaa=dataa){
  return (mean(bin(x,binsize,data=dataaa)[,2]))
}
```

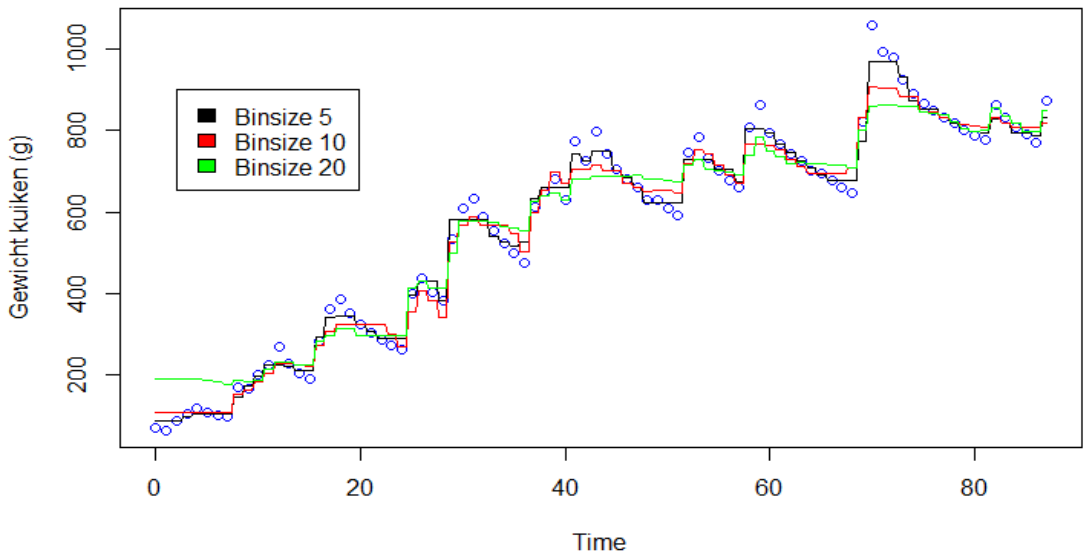
met een paar extra lijntjes kunnen we een model opstellen voor *Bird1*

```
X = seq(0,87,0.2)
Y=c()
for (i in 1:length(X)){
  Y=c(Y , bin_smooth(X[i],20))
}
lines(X,Y,col="green")
```

We krijgen volgende grafieken.
We zien duidelijk dat een grotere *binsize* ons een minder oscillerend, maar stabieler model geeft omdat het meer punten samen pakt om een gemiddelde te berekenen.


binsize=5
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


binsize=20
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


Binsize 5
Binsize 10
Binsize 20
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

5 Lowess

Lowess staat voor *Local Weighted Regression*. Het lijkt sterk op simpele lineaire regressie, met het de extra factor van een *weight* per datapunt. Dit kan handig zijn als je variantie toeneemt met de regressor. Dan kan je punten met meer variantie een kleiner gewicht geven zodat deze minder zouden door tellen. Dit is ook duidelijk bij het concept van *weighted linear regression*. Bij simpele regressie probeert men de Sum Squared Error te minimaliseren met volgende formule:

$$SSE(\hat{\beta}) = \sum_{i=1}^{n} (Y_i - m(x_i; \hat{\beta}))^2 = \|\mathbf{Y} - \mathbf{X}\hat{\beta}\|^2$$

Bij gewogen regressie is dit een gelijkaardige formule met de extra term voor de gewichten:

$$WSSE(\hat{\beta}) = \sum_{i=1}^{n} w_i (Y_i - m(x_i; \hat{\beta}))^2$$

Theorem(Weighted Least Squares) *Veronderstel dat* $\boldsymbol{X}$ *de design matrix is met rang p.* $W \in \mathbb{R}^{n \times n}$ *de weight matrix, een diagonaal matrix met de gewichten* w_i *op de diagonaal. Dan wordt de Weighted Least Squares van* β *gegeven door*

$$\hat{\beta} = (X^T W X)^{-1} X^T W Y$$

en deze oplossing is uniek.

Proof. We zoeken een β die $WSSE(\beta) = \sum_{i=1}^{n} w_i (Y_i - m(x_i; \beta))^2$ minimaliseert. Dus

$$\beta = ArgMin_\beta WSSE(\beta)$$

Verder kunnen we $WSSE(\beta)$ schrijven als volgt:

$$WSSE(\beta) = (\mathbf{Y} - \mathbf{X}\beta)^T \mathbf{W} (\mathbf{Y} - \mathbf{X}\beta)$$

Als we dit distributief uitwerken krijgen we:

$$WSSE(\beta) = (\mathbf{Y}^T \mathbf{W} \mathbf{Y} - \mathbf{Y}^T \mathbf{W} \mathbf{X} \beta - \beta^T \mathbf{X}^T \mathbf{W} Y + \beta^T \mathbf{X}^T \mathbf{W} \mathbf{X} \beta)$$

De weighted least squares schatter voldoet dan aan:

$$\begin{aligned}
&\frac{d}{d\beta} SSE(\beta) = 0 \\
&\qquad \Updownarrow \\
&2X^T W X\beta = 2X^T WY \\
&\qquad \Updownarrow \\
&\hat{\beta} = (X^T WX)^{-1} X^T WY
\end{aligned}$$

Verder moet de 2de partiële afgeleide positief definitief zijn zodat we zeker een minimum hebben.

$$\frac{d}{d\beta} 2(X^T WX\beta - X^T WY) = 2X^T WX$$

Omdat W een diagonaal matrix is met positieve gewichten ($w_i \in [0, 1]$), is W positief definitief. Hier uit geldt dat $X^T WX$ ook positief definitief is.
$\forall v \in \mathbb{R}^n$:

$$v^T(X^T WX)v = (v')^T W(v') > 0 \qquad \text{want } W \text{ is positief definitief}$$

De uniciteit wordt bewezen op dezelfde manier als bij simpele lineaire regressie. Stel we hebben een $\hat{\beta}_1$ en een $\hat{\beta}_2$ met $\hat{\beta}_1 \neq \hat{\beta}_2$ en $\hat{\beta} = (X^T WX)^{-1} X^T WY$ met dan $(X^T WX)$ positief definitief geldt:

$$\begin{aligned}
(X^T WX)\hat{\beta}_1 &= (X^T WX)\hat{\beta}_2 \\
&\Updownarrow \\
(X^T WX)(\hat{\beta}_1 - \hat{\beta}_2) &= 0 \\
&\Updownarrow \\
(\hat{\beta}_1 - \hat{\beta}_2) &= 0 \\
&\Updownarrow \\
\hat{\beta}_1 &= \hat{\beta}_2 \quad \lightning
\end{aligned}$$

□

We kunnen nu *Local Weighted Regression* met vertrouwen gebruiken om een model op te stellen voor onze vogeltjes. Als gewichts functie gebruiken we de *Tukey's tri-cube weight function* die waarden geeft afhankelijk van hoe dicht ze staan bij ons gegeven punt, met dichtbij naar 1 gaande en ver naar 0. Deze weight functie voor een *neighbourhood* rond x_k wordt gegeven door:

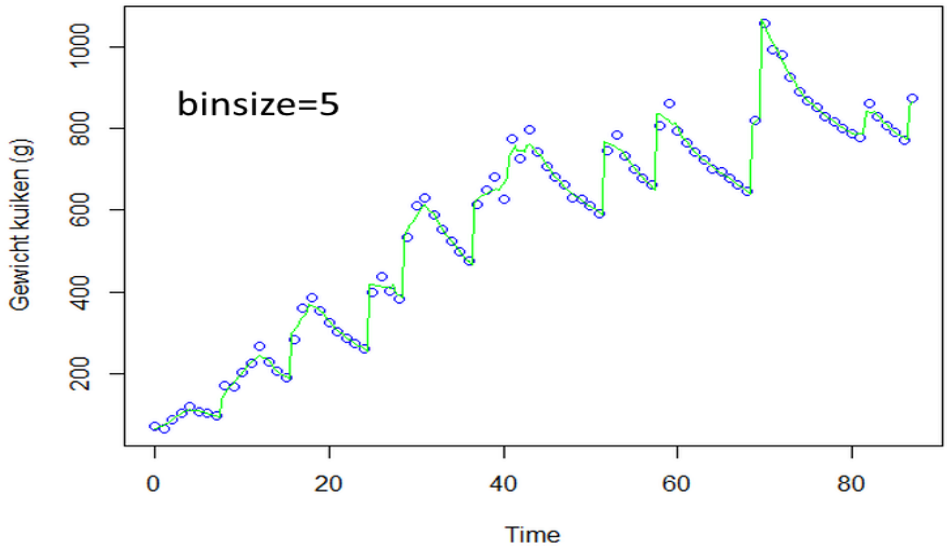
$$w_k(x_i) = (1 - |(x_i - x_k)/h|)^3$$

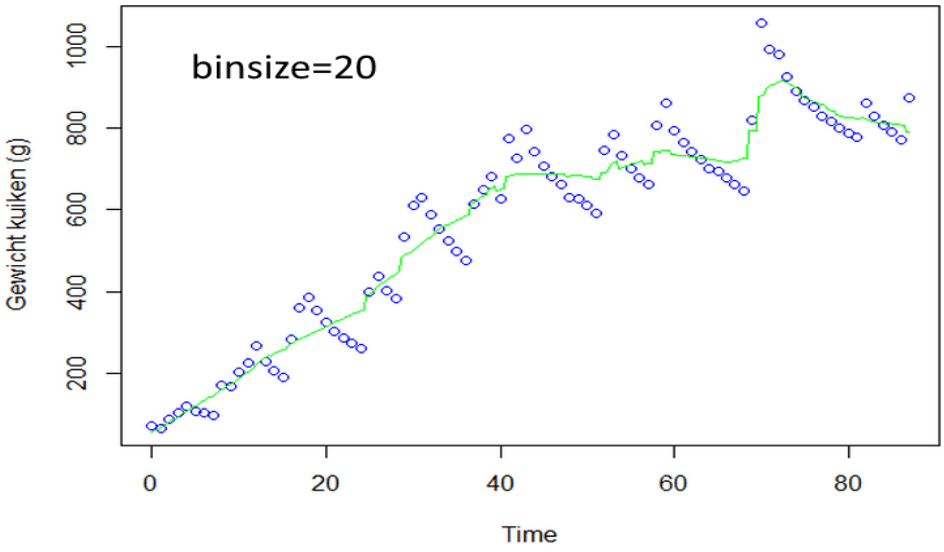
Waarbij h de bandbreedte is van een neighbourhood zodat de weights tussen 0 en 1 liggen.

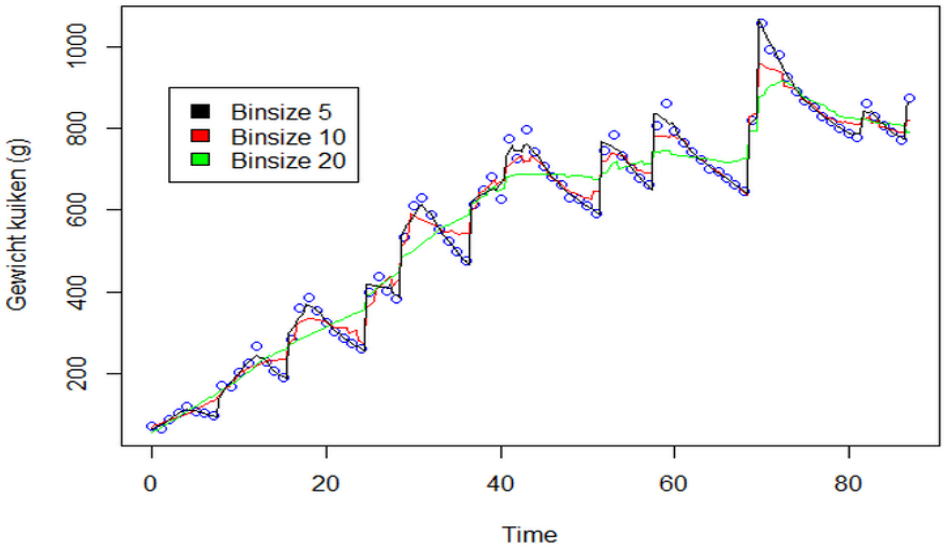
Hier heb ik weer code voor geschreven: een `weight` en `lowess` functie, waarbij zoals hierboven beschreven *weighted regression* wordt toegepast.

```
weight = function(binn,x,x0){
  h=0
  for (i in 1:nrow(binn)){
    d = abs(binn[i,1]-x0)
    h=max(c(h,d))
  }
  return((1 - abs((x-x0)^3/h^3))^3)
}

lowess  = function(x,binsize,data=dataa){
  binn = bin_band(x,binsize,data=dataa)
  X = cbind(matrix(rep(1,nrow(binn))),binn[,1])
  Y = matrix(binn[,2])
  W=c()
  for (i in 1:nrow(binn)){W=c(W,weight(binn,binn[i,1],x))}
  W=matrix(diag(W),ncol=length(W))
  B = solve(t(X)%*%W%*%X)%*%t(X)%*%W%*%Y
  return(matrix(c(1,x),nrow=1)%*%B)
}
```

en met extra lijntjes om terug te plotten zoals bij Bin Mean, krijgen volgende resultaten:


binsize=5
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


binsize=20
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

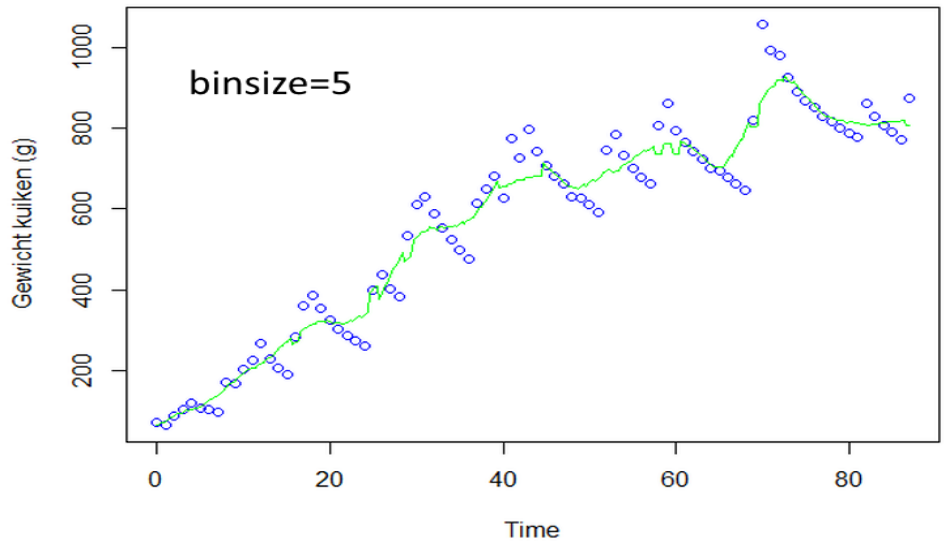

Binsize 5
Binsize 10
Binsize 20
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

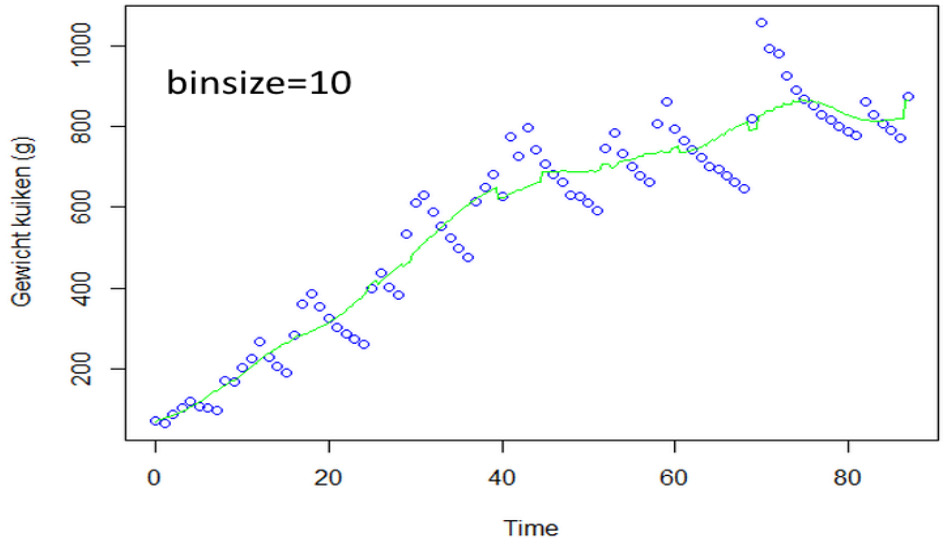
Deze resultaten zijn niet zo *smooth* als we zouden verwachten. Uit bronnen kan men vernemen dat vaak ook *bins* worden gemaakt met een bepaalde *bandwidth*, en dus niet met een vaste grootte. Na verdere inspectie blijkt dat deze *width* ook afhankelijk is van de dichtste datapunt. Dit kan een manier zijn om een gladder model te krijgen Met volgende R code stellen we opnieuw een *bin* op maar met een bepaalde *bandwidth* die afhankelijk is van het dichtste punt, namelijk:

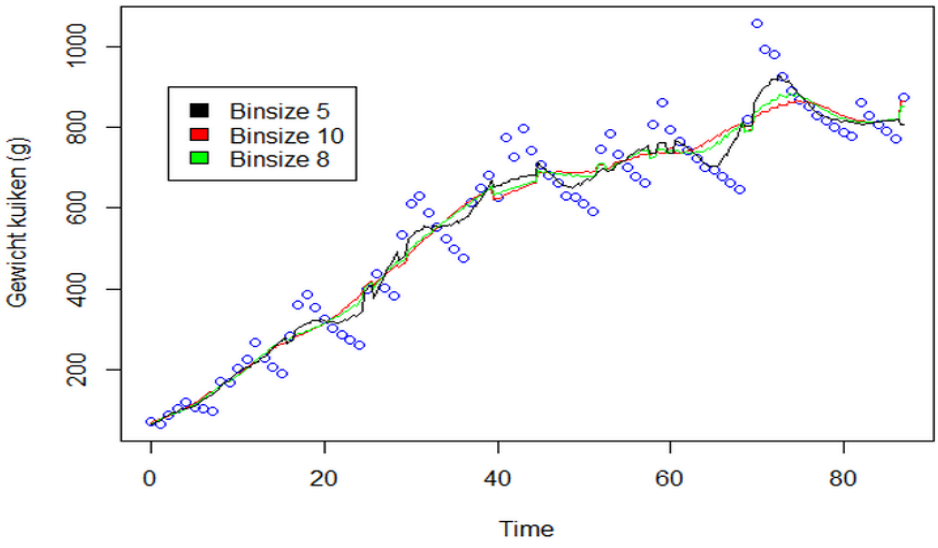
$$band\_width(x) = (closest\_point_X - x) * binsize$$

waarbij *binsize* wordt gebruikt om een bin op te stellen waar daaruit het dichtste punt wordt genomen.

```
bin_band = function(x,binsize,data=dataa){
  x = round(x)
  x_min = x-binsize
  if(x_min<0){x_min=0}
  x_max = x+binsize
  if(x_max>dim(data)[1]){x_max=dim(data)[1]}
  #options for bin
  neighbor_opt = data[x_min:x_max,1:2]#BIRD 1 == INDEX 2
  for (i in 1:nrow(neighbor_opt)) {
    dis = sqrt( ((neighbor_opt[i,1]-x)/(x+as.integer(x==0)))^2+
    ...((neighbor_opt[i,2]-data[x+1,2])/data[x+1,2])^2)
    neighbor_opt[i,3] = dis
  }
  neighbor_opt = neighbor_opt[neighbor_opt[,1]!=x,]
  closest = neighbor_opt[neighbor_opt[,3] == min(neighbor_opt[,3]),1:2]
  width = abs(closest[1,1]-x)*binsize
  lowX = x-width
  UpX = x+width
  if(lowX<0){lowX=0}
  if(UpX>88){upX = 88}
  binn = data[lowX:UpX,1:2]
  binn = na.omit(binn)
  return (binn)
}
```


binsize=5
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


binsize=10
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


Binsize 5
Binsize 10
Binsize 8
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

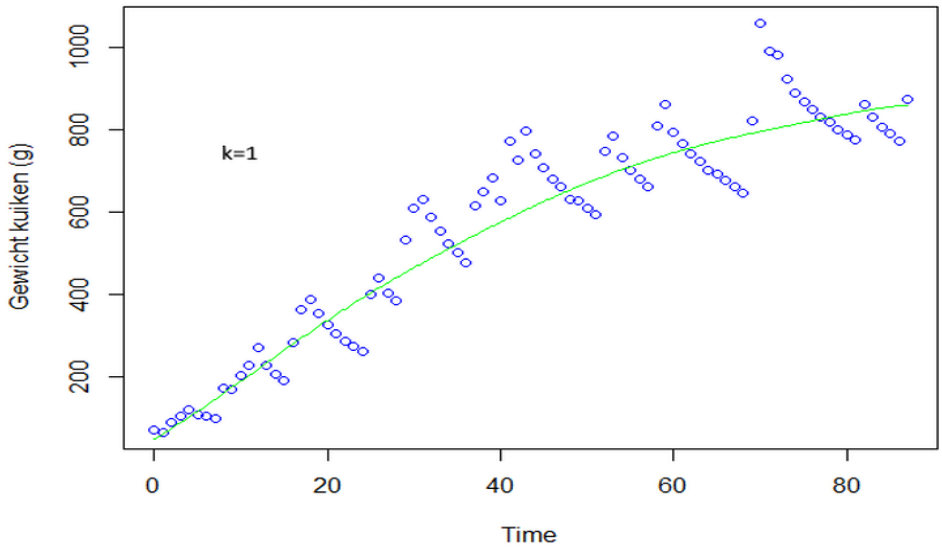
We zien opnieuw dat het model nog altijd een beetje *krokant* is. Het wordt wel sneller 'gladder' met een hogere *binsize*.
Al deze niet gladde kronkels kunnen te verklaren zijn doordat we ten eerste 'maar' lineair werken op een *bin*, terwijl veeltermen ook altijd een optie is. En dat we gewoon werken met *bins* waar variabelen binnen en buiten vallen en zo harde veranderingen kunnen doordrijven.
Een laatste verbetering die we misschien kunnen maken is om niet met *bins* te werken, maar gewoon elk datapunt een gewicht te geven afhankelijk van uw gekozen X. Om een *smoothing*-factor te behouden definiëren we onze *weight* functie als een variatie van de *tri-cube weight* functie met een extra factor k die bepaalt hoe snel de gewichten naar 0 gaan, naar buiten gaande, en dus gaat dit een gladder of niet gladder geheel geven. Als k naar 0 gaat worden de gewichten groter voor buitenstaande data, en wordt het model dus gladder. Als k naar ∞ gaat dan worden de gewichten sneller 0 en gaat het model terug meer naar de datapunten trekken.

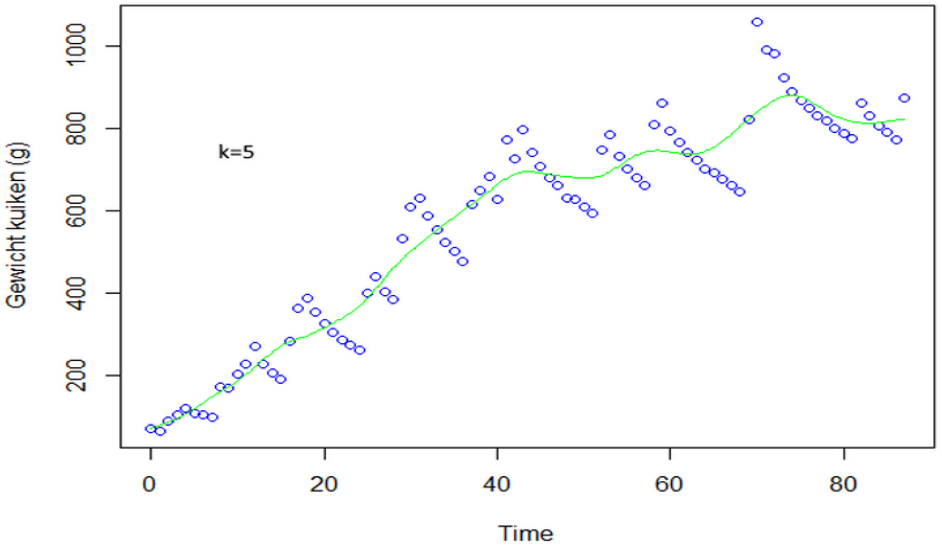
$$w_{x_0}(x) = \begin{cases} (1-((x-x_0)*k/h)^3)^3 & \text{als } (1-((x-x_0)*k/h)^3)^3 > 0 \\ 0 & \text{als } (1-((x-x_0)*k/h)^3)^3 < 0 \end{cases}$$

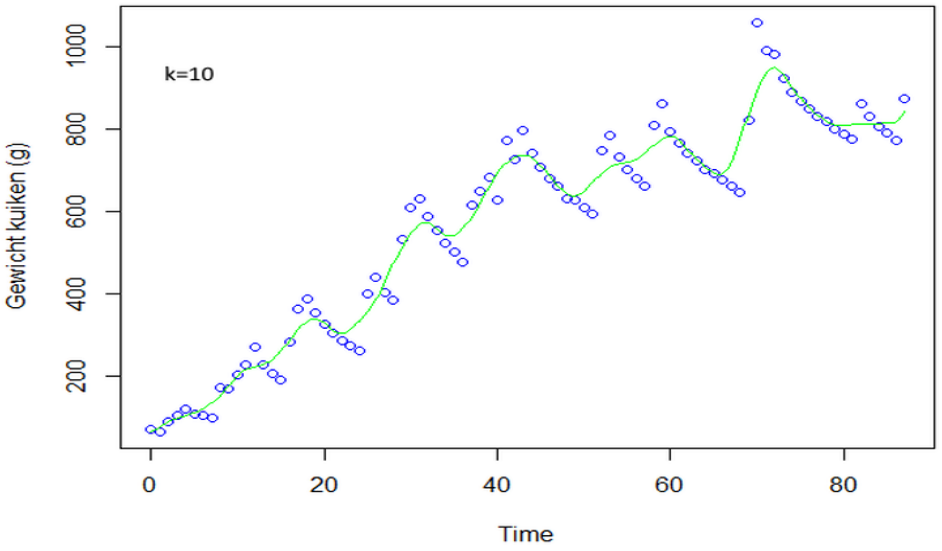
We zullen h gelijk stellen aan 45 omdat onze data $89 \approx 90$ datarijen bevatten, en dus de helft van ons venster ongeveer 45 is. Met een nieuwe functie kunnen opnieuw wat modelletjes maken.

```
lowess2  = function(x,k,data=dataa){
  X = cbind(matrix(rep(1,nrow(data))),data[,1])
  Y = matrix(data[,2])
  W=c()
  for (i in 1:nrow(data)){
    if((1 - abs((data[i,1]-x)*k/45)^3)^3<0){
      W=c(W,0)
    }
    else{
      W=c(W, (1 - abs((data[i,1]-x)*k/45)^3)^3)
    }
    }
  W=matrix(diag(W),ncol=length(W))
  B = inv(t(X)%*%W%*%X)%*%t(X)%*%W%*%Y
  return(matrix(c(1,x),nrow=1)%*%B)
```

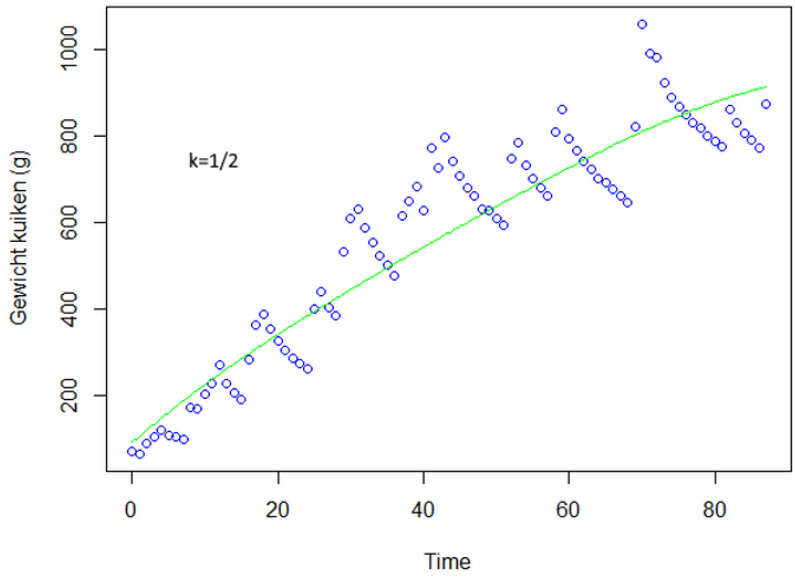
```
}
```

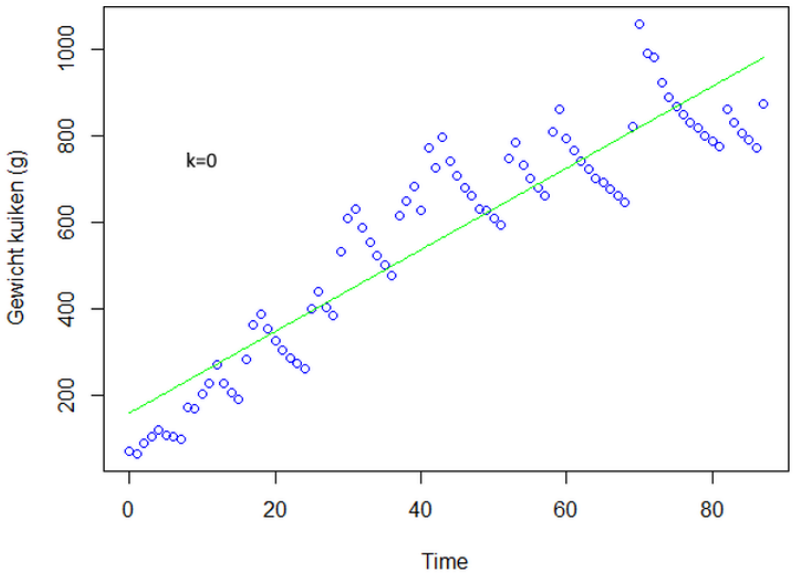

k=1
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


k=5
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


k=10
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

Dit is een hele verbetering. Het model ziet er perfect glad uit. We zien ook duidelijk dat als k naar 0 gaat dat het dus gladder wordt, en rechter. We zien duidelijk in volgende grafieken dat als $k = 0$ dat we gewoon een lineair model krijgen.





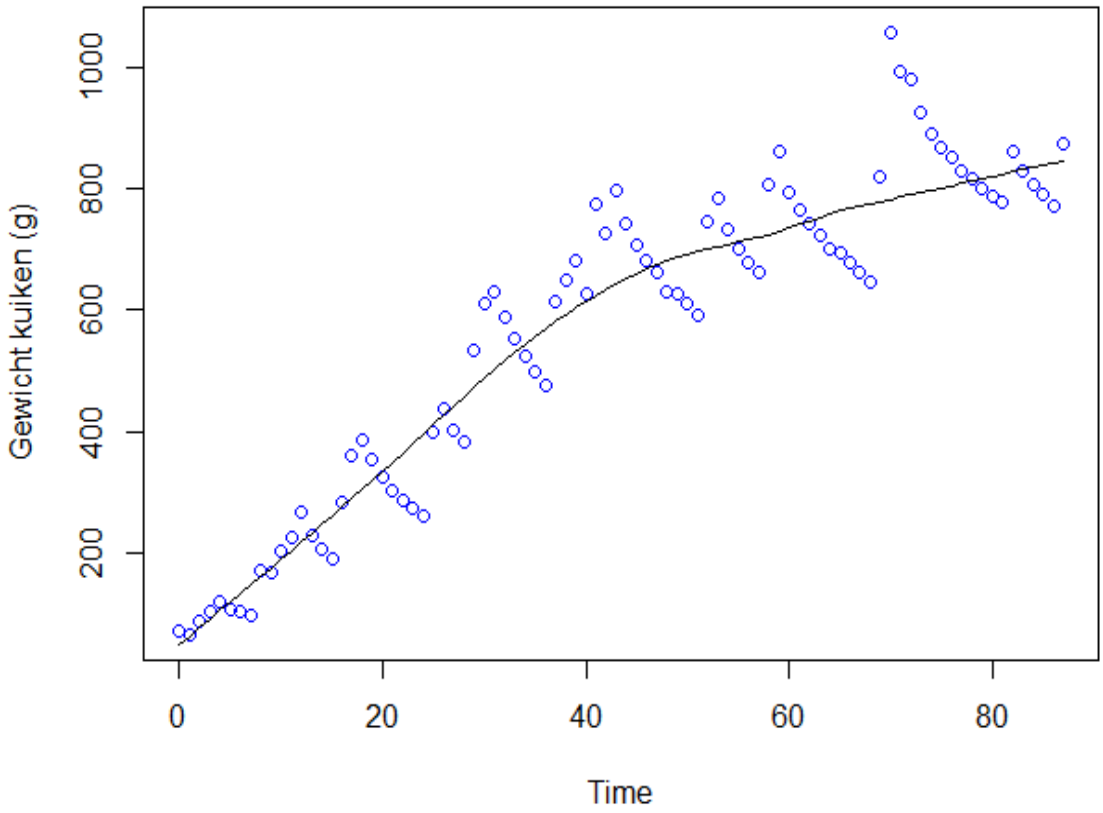
Als $k = 0$ zien we zelfs dat dit hetzelfde lineair verband is die we kregen met de introductie. Er zijn dus duidelijk grenzen voor k waar binnen het een goed

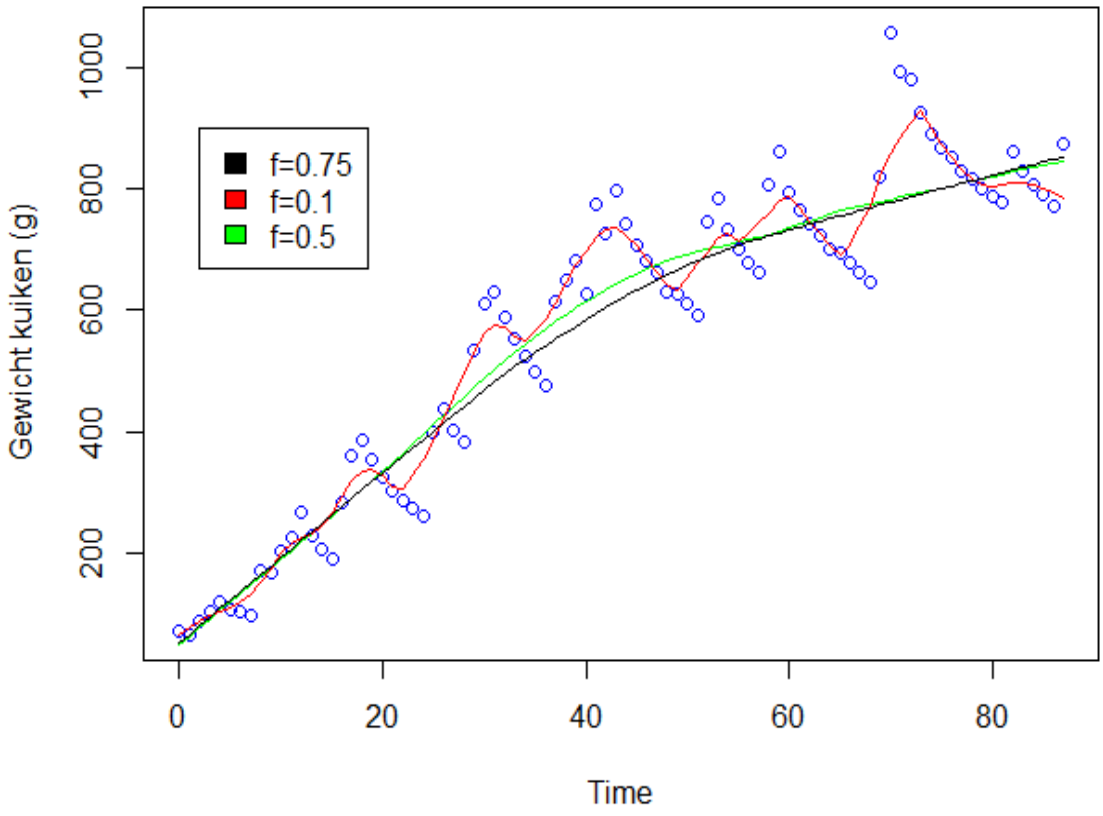
model is en buiten een niet zo goed model. Te groot, en we krijgen verbonden datapunten, te klein en we krijgen een lineair verband dat ook niet goed was. Let op voor te grote waarden voor k omdat deze een negatieve gewicht kan geven, en dus 0 wordt. Of door computernauwkeurigheid, afrondingen naar 0 kan geven. Het resultaat is dat de matrix W niet meer inversiebel is omdat deze een 0 rij bevat en er dus geen oplossing bestaat voor die gegeven x

`R` heeft zelf ook een `lowess` functie. De functie ontvangt een variabele `f= 2/3` die de gladheid van de smoother aanpast. Als `f` naar 0 gaat, wordt het model gewoon op de datapunten geplot, met `f` gaande naar 1 wordt het model gladder en gladder en dus platter en platter. verder *returnt* deze functie 2 vector `x` en `y` die gewoon de punten van het model bevatten.

```
> m=lowess(dataa$Time,dataa$Bird1,f=0.1)
> head(m$x)
[1] 0 1 2 3 4 5
> head(m$y)
[1]  65.43573  76.83904  87.61375  97.85081 104.32112 109.60283
>
```

De `lowess` functie is hier dus NIET hetzelfde als mijn eigen functie, maar de standaard `R` functie.


Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time


f=0.75
f=0.1
f=0.5
Gewicht kuiken (g)
200
400
600
800
1000
0
20
40
60
80
Time

6 Predicties

Het lijkt nu moeilijk om predicties te doen met dit model buiten de gegeven data en dus buiten het model (extrapolatie). Om nu predicties te maken met een smooth model wordt er in bronnen gedaan aan het interpoleren van de verkregen X en Y punten van het model om zo een veelterm functie op te stellen zodat je punten gaat kunnen voorspellen.

References

[1] *Chapter 5, Nonparametric Regression.* Universidad Carlos III de Madrid.

[2] *Lecture 24: Weighted and Generalized Least Squares.* Carnegie Mellon University, Statistics & Data Science.

[3] S. Dobilas. Lowess regression in python: How to discover clear patterns in your data?

[4] prof. Rafael Irizarry. *Chapter 3, Local Regression.* Harvard University.

[5] Statology/Zach. How to perform lowess smoothing in r.

[6] *MikeLove* on Github. *Data Analysis for Genomics.* 2014.

[7] D. o. S. S. Trinity College. Smoothing, lectures 218.