# Grenzen voor de simulator

## Inleiding

Voor deze opdracht was het de bedoeling om een boven- en ondergrens te bepalen waaraan de resultaten van de simulator moeten voldoen door de simulator verschillende keren te laten lopen met verschillende random seeds.

## Antwoord

We hebben twee verschillende manieren gebruikt om de grenzen te bepalen.

De eerste en eenvoudigste manier is om te kijken naar de gemiddelde waarde en daar een marge van 10% rond te nemen. Hiervoor hebben we een aantal histogrammen gemaakt die kunnen worden teruggevonden in bijlage. Hieruit blijkt dat het gemiddelde altijd ligt rond 106700. Door hier een marge van 10% boven en onder te nemen krijgen we als afgeronde waarden 96000 voor de ondergrens en 117400 voor de bovengrens. De echte waarden kunnen worden gevonden bij de histogrammen.

Een tweede manier om deze grenzen te bepalen is door een interval op te stellen zodanig dat de kans dat een resultaat binnen deze grenzen valt praktisch 100% is. Hieruit halen we dan een ondergrens van grofweg 104000 en een bovengrens van 109000. Dit geeft respectievelijk ongeveer 2.53% en 2.16% afwijking rond het gemiddelde. Indien we dit afronden naar boven en 3% nemen krijgen we grenzen die ongeveer gelijk zijn aan 103500 en 109900.

Hierbij zouden we willen opmerken dat de waarde voor de random seed niet echt van belang is: zoals op de point plots in bijlage te zien valt is het aantal geïnfecteerden quasi gelijk verdeeld over de verschillende seed waarden.

Ook hebben we gemerkt dat, ongeacht de manier waarop de seeds gekozen worden, de resultaten een normale verdeling blijven volgen. Dit wordt bevestigd door zowel de verschillende plots als uitvoering van de Shapiro-Wilk test.

Als laatste zouden we willen meegeven dat de resultaten die we zijn bekomen alleen gelden voor measles aangezien dit de default instelling was. Andere ziektes kunnen andere resulaten geven.

## Werkwijze

Om deze grenzen te bepalen zijn we als volgt tewerk gegaan.

Als eerste hebben we naar een manier gezocht waarop we de simulator voor verschillende random seeds konden laten werken. Dit kan worden teruggevonden in seedTester.py: door in een terminal `python3 seedTester.py <path-to-file-with-seeds> <num-of-days>` uit te voeren, waar beide argumenten optioneel zijn, worden voor alle random seeds in `<path-to-file-with-seeds>` een simulatie van `<num-of-days>` dagen (standaard 50) uitgevoerd waarvan het laatste resultaat telkens wordt weggeschreven.

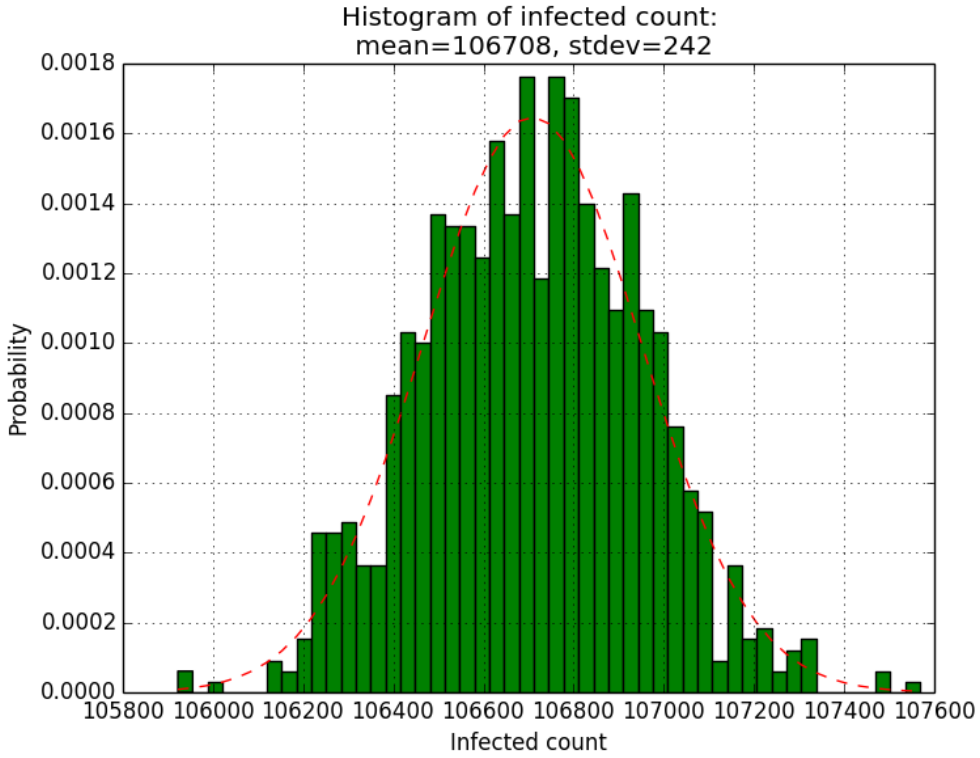
Na de resultaten te hebben verzameld zijn we aan de slag gegaan met de verwerking ervan. Om te beginnen hebben we histogrammen gemaakt van de data om een eerste indruk te krijgen van hoe de data verdeeld is. Dit gaf ons een eerste indicatie dat de data een normale verdeling volgde. Om dit vermoeden te bevestigen hebben we nadien ook gekeken naar de QQ-plots en boxplots. Beide bevestigden ons vermoeden van een normale verdeling. Als laatste hebben we de Shapiro-Wilk test uitgevoerd om helemaal zeker te zijn. Wederom bevestiging van de normale verdeling. Tot slot bepalen we hieruit grenzen zoals aangegeven in het vorige hoofdstuk.

## Conclusie

We concluderen dus dan we 3% rond het gemiddelde kunnen nemen terwijl we met zeer grote zekerheid kunnen zeggen dat alle resultaten binnen dit interval zullen vallen. Een conservatievere waarde zou eventueel 5% kunnen  zijn om iets meer marge toe te laten aangezien we slechts 5000 simulaties hebben uitgevoerd in contrast tot de 4,294,967,295 mogelijke seeds.

# Bijlage

## Histogrammen [1]

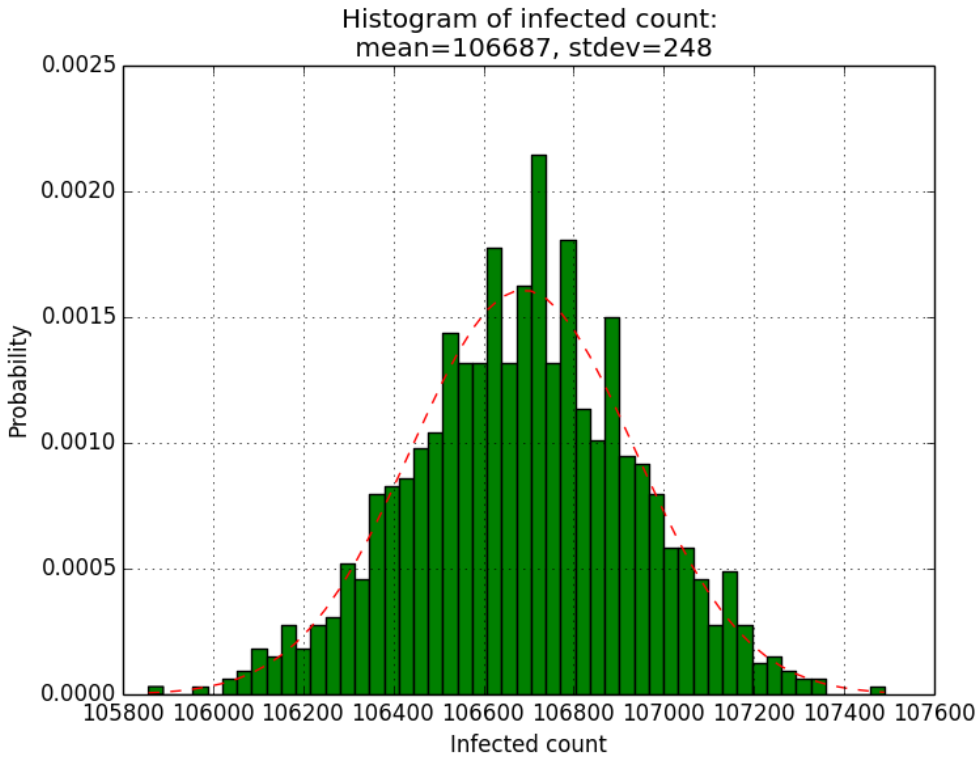



Histogram van lineair gegenereerde seeds (linear.dat).

Shapiro-Wilk test: W = 0.99803, p = 0.2959 => voor alpha=0.05 nulhypothese niet verwerpen
=> data normaal verdeeld

| **Statistiek** | **plotter.py** | **R** |
|---|---|---|
| Gemiddelde | 106708 | 106708.2 |
| Std. Deviatie | 242 | 242.3362 |
| 10% Grenzen | [96037,117379] | |
| [Minimum, Maximum] | | [105922,107568] |
| P(Minimum < X < Maximum) | | 99.92169% |

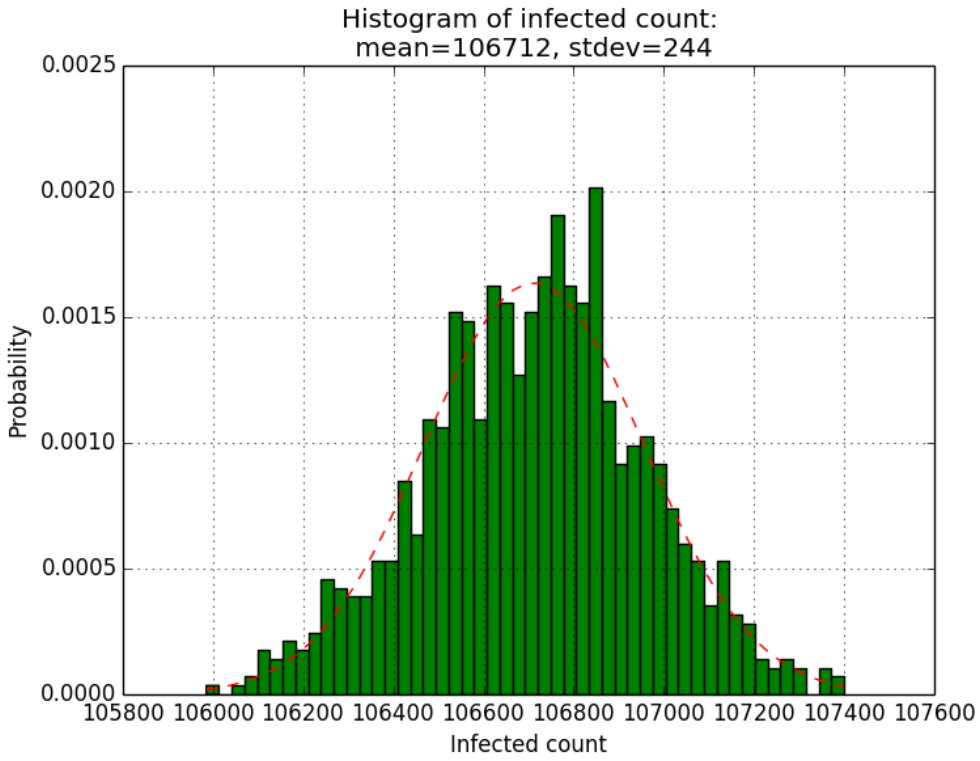
1 Iedere balk in de histogrammen geeft aan hoe vaak die bepaalde klasse is voorgekomen bij alle verschillende random seeds.




Histogram van lineaire seeds met ruis (noise1.dat)

Shapiro-Wilk test: W = 0.99919, p = 0.9515 => voor alpha=0.05 nulhypothese niet verwerpen
=> data normaal verdeeld

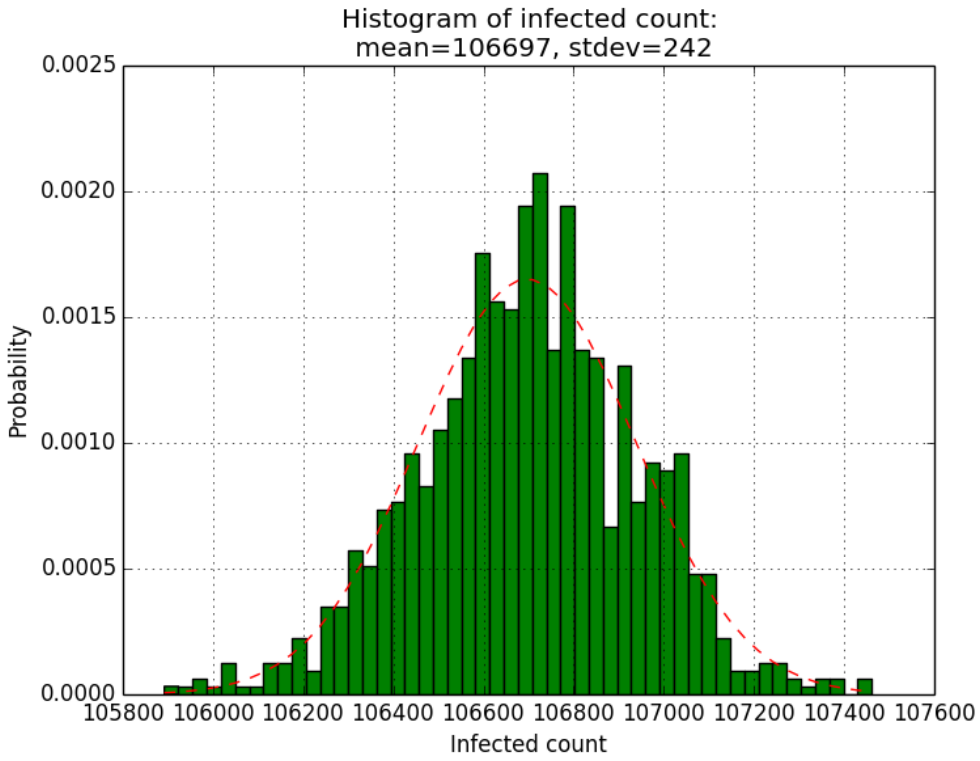
| **Statistiek** | **plotter.py** | **R** |
|---|---|---|
| Gemiddelde | 106687 | 106687.3 |
| Std. Deviatie | 248 | 248.1784 |
| 10% Grenzen | [96019,117356] | |
| [Minimum, Maximum] | | [105857,107489] |
| P(Minimum < X < Maximum) | | 99.89712% |




2e histogram van lineaire seeds met ruis (noise2.dat).

Shapiro-Wilk test: W = 0.99791, p = 0.2479 => voor alpha=0.05 nulhypothese niet verwerpen
=> data normaal verdeeld

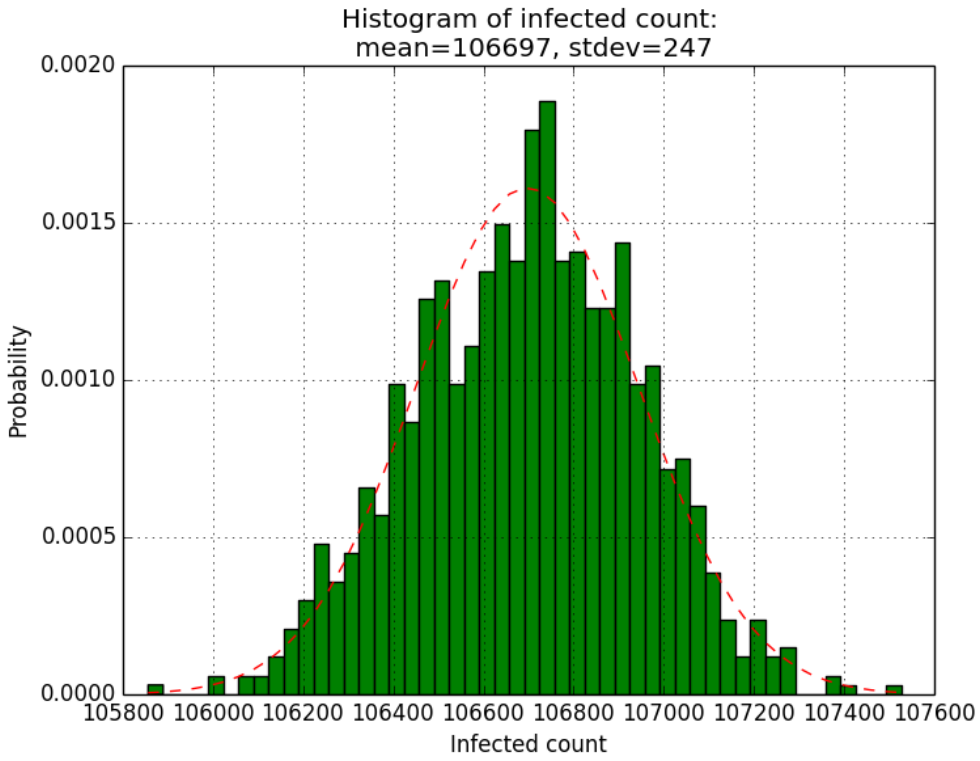
| **Statistiek** | **plotter.py** | **R** |
|---|---|---|
| Gemiddelde | 106712 | 106712 |
| Std. Deviatie | 244 | 243.9047 |
| 10% Grenzen | [96041,117383] | |
| [Minimum, Maximum] | | [105985,107400] |
| P(Minimum < X < Maximum) | | 99.61664% |




Histogram van random gegenereerde seeds (random1.dat).

Shapiro-Wilk test: W = 0.99778, p = 0.2014 => voor alpha=0.05 nulhypothese niet verwerpen
=> data normaal verdeeld

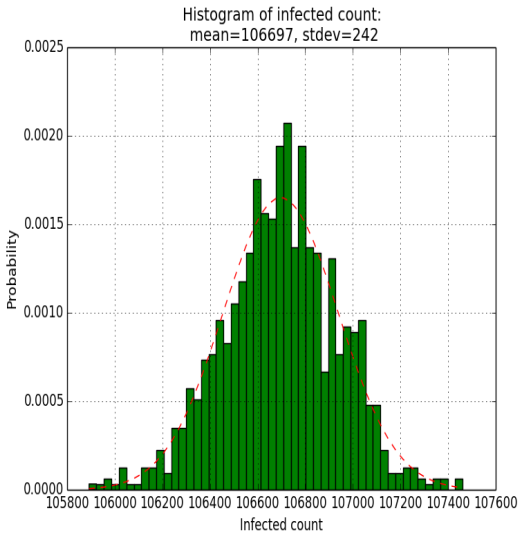
| **Statistiek** | **plotter.py** | **R** |
|---|---|---|
| Gemiddelde | 106697 | 106696.9 |
| Std. Deviatie | 242 | 241.7712 |
| 10% Grenzen | [96027,117367] | |
| [Minimum, Maximum] | | [105892,107461] |
| P(Minimum < X < Maximum) | | 99.87769% |



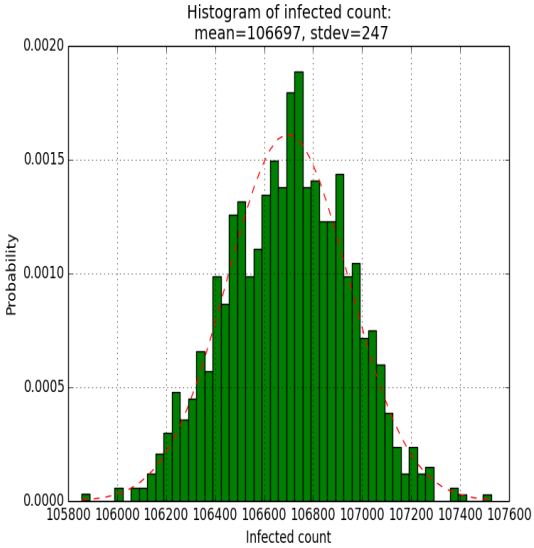

2e histogram van random gegenereerde seeds (random.dat).

Shapiro-Wilk test: W = 0.99807, p = 0.3144 => voor alpha=0.05 nulhypothese niet verwerpen
=> data normaal verdeeld

| **Statistiek** | **plotter.py** | **R** |
|---|---|---|
| Gemiddelde | 106697 | 106697.1 |
| Std. Deviatie | 247 | 247.6202 |
| 10% Grenzen | [96027,117367] | |
| [Minimum, Maximum] | | [105856,107526] |
| P(Minimum < X < Maximum) | | 99.92513% |

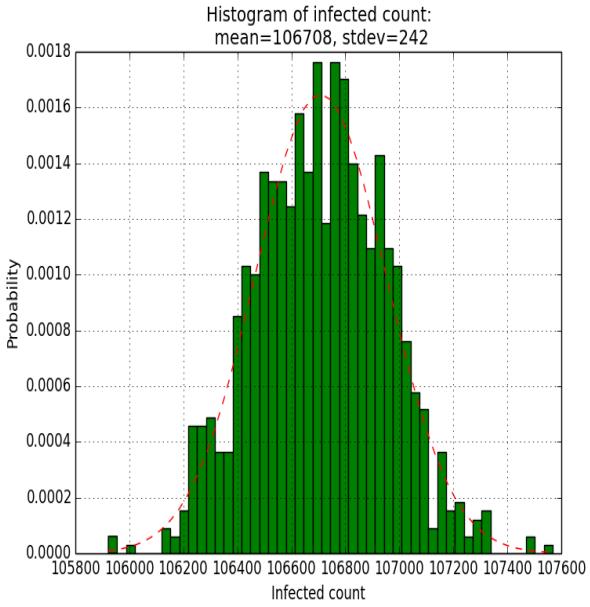



random1.dat



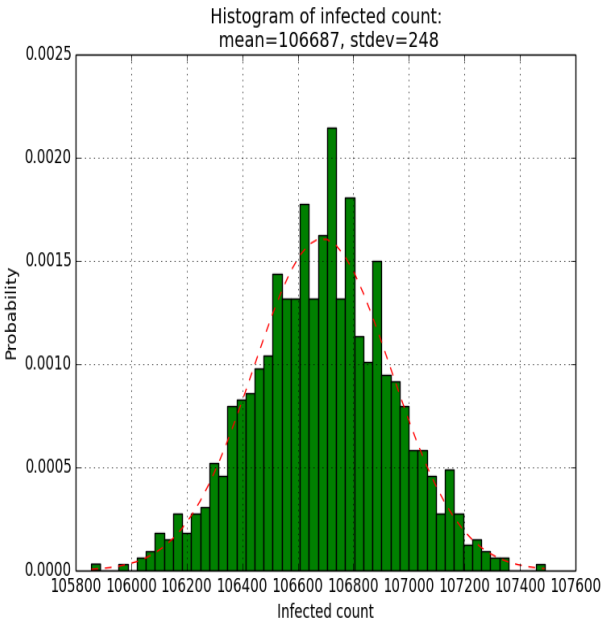

linear.dat

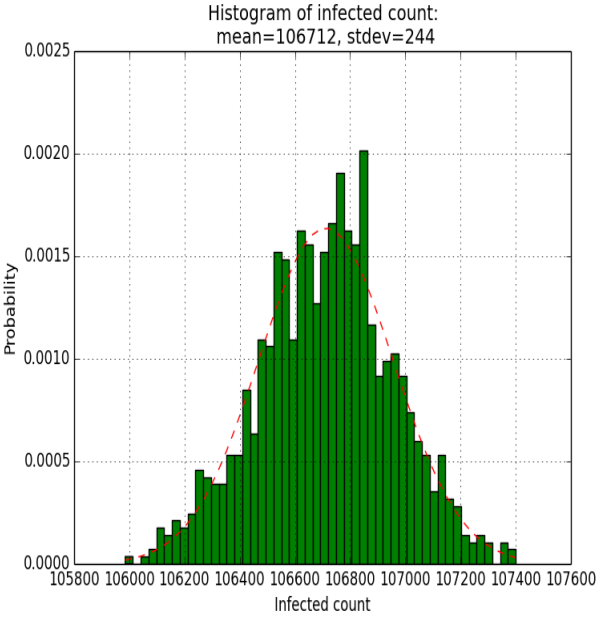
random2.dat



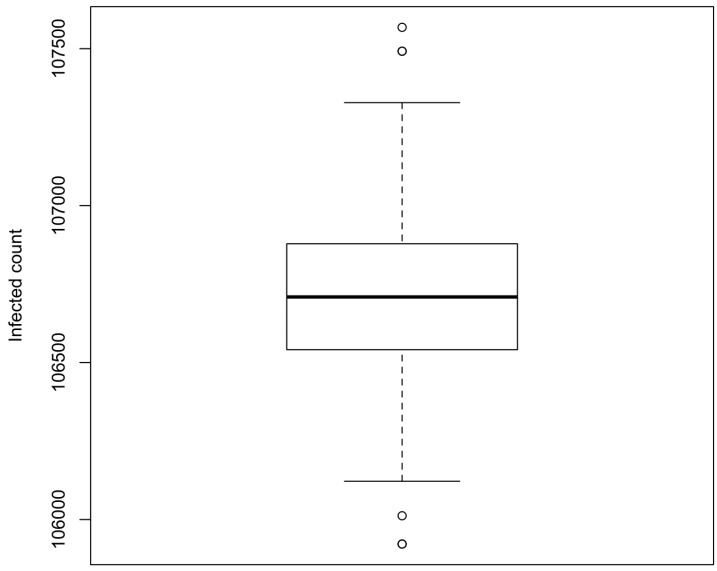

noise1.dat

noise2.dat

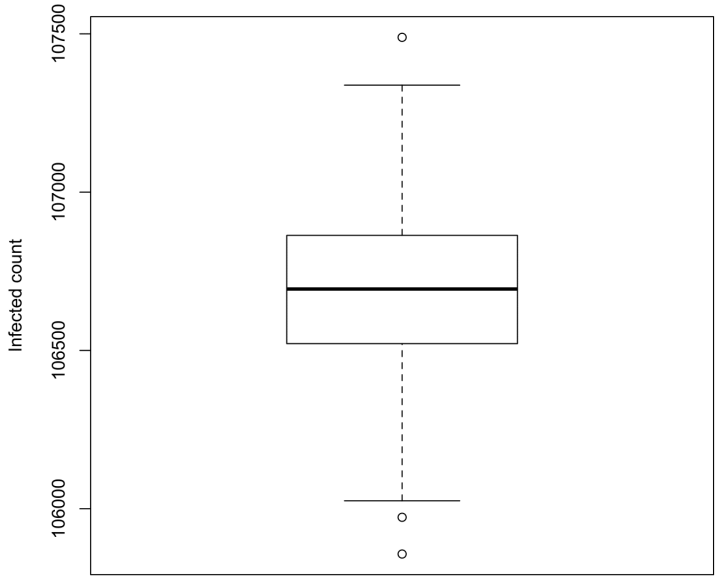


## Boxplots [2]

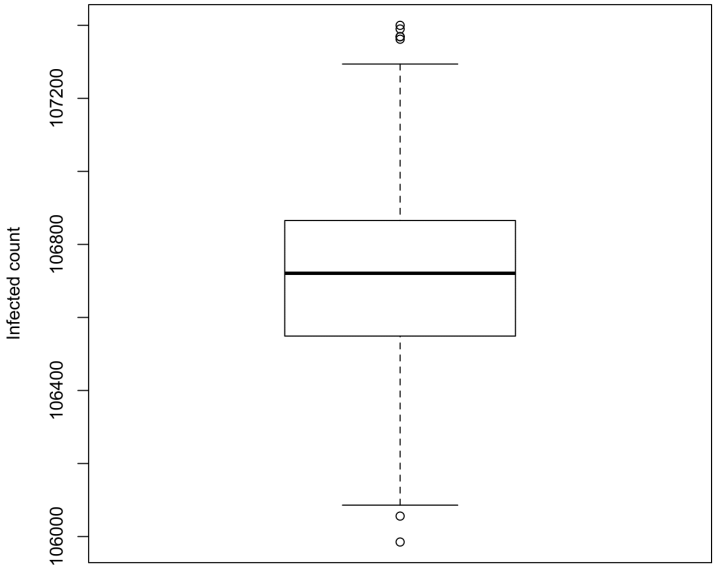



Boxplot voor lineair gegenereerde seeds.

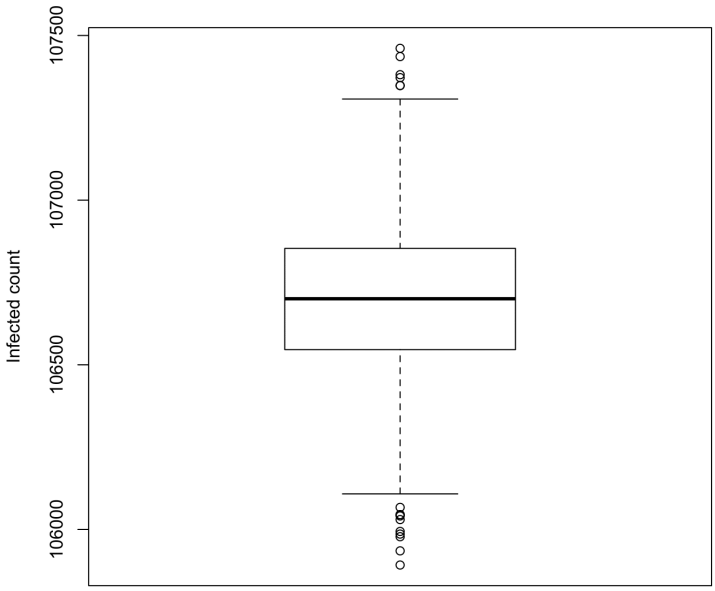



Boxplot voor lineaire seeds met ruis.

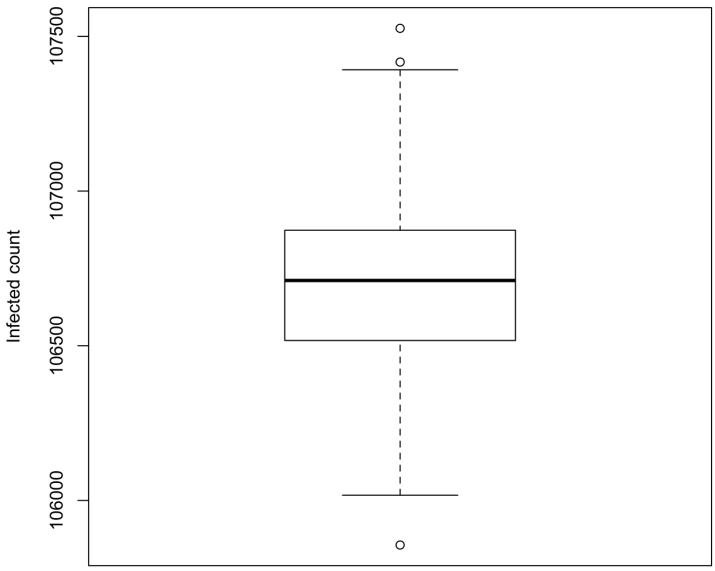
2 Ieder deel tussen 2 horizontale strepen bevat een even groot percentage van de data.



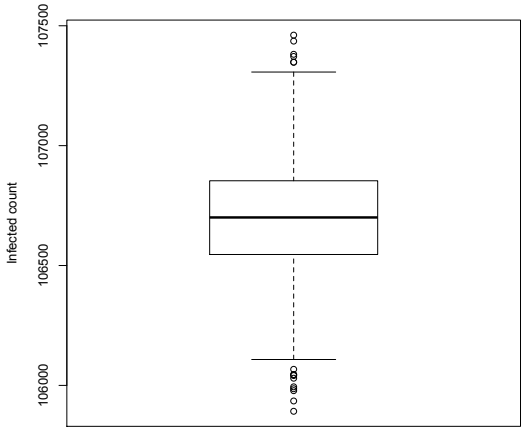

2e boxplot voor lineaire seeds met ruis.



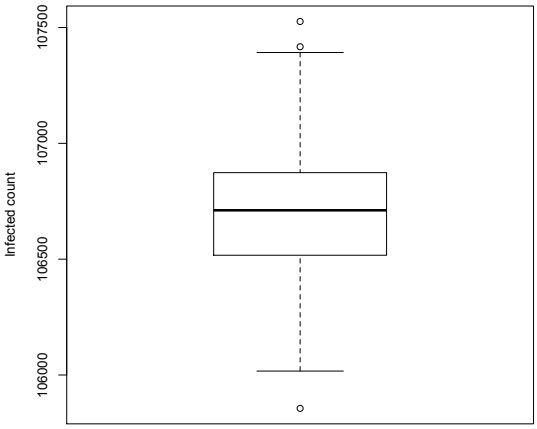

Boxplot voor random gegenereerde seeds.




2e boxplot voor random gegenereerde seeds.

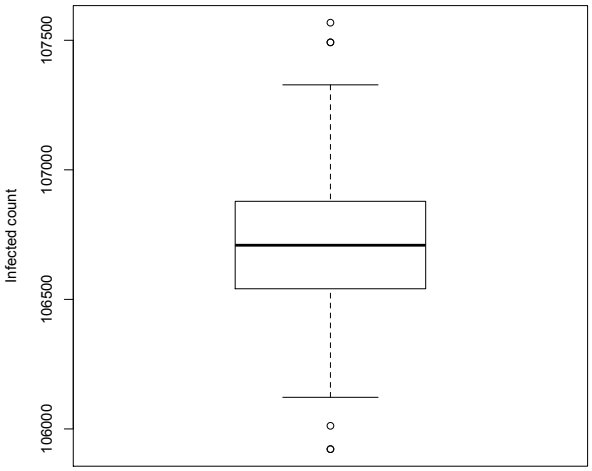



random1.dat



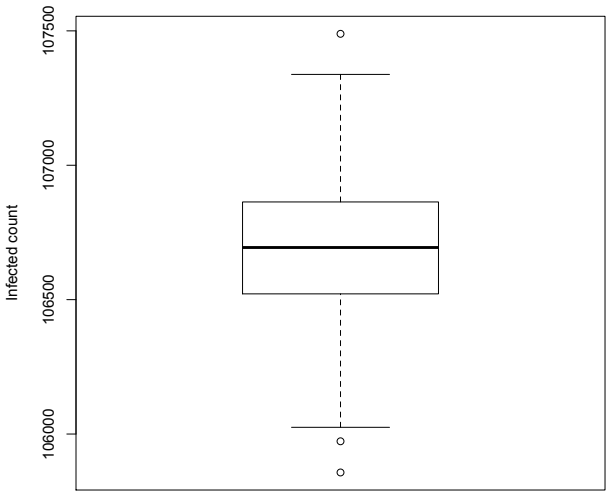

linear.dat

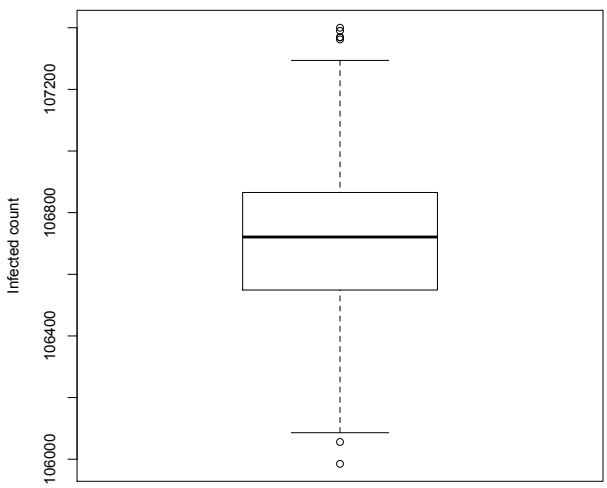
random2.dat



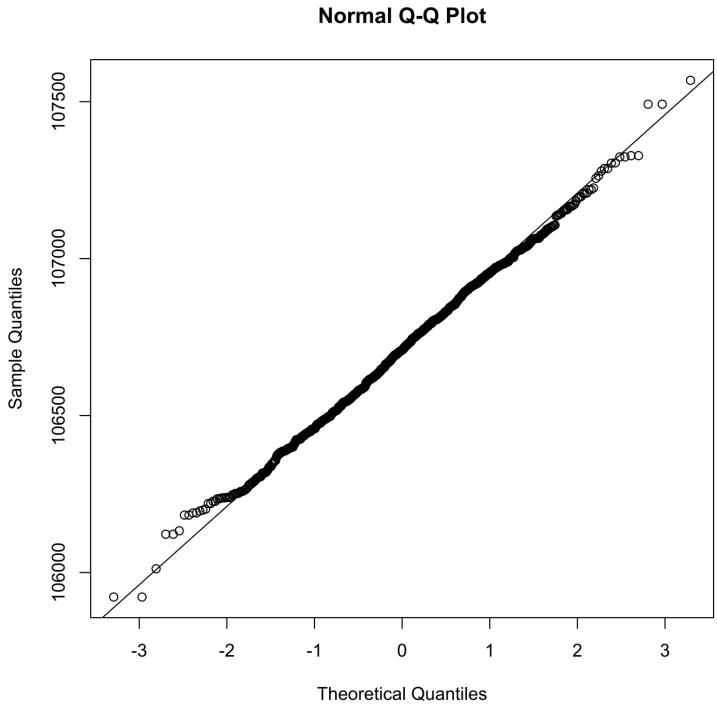

noise1.dat

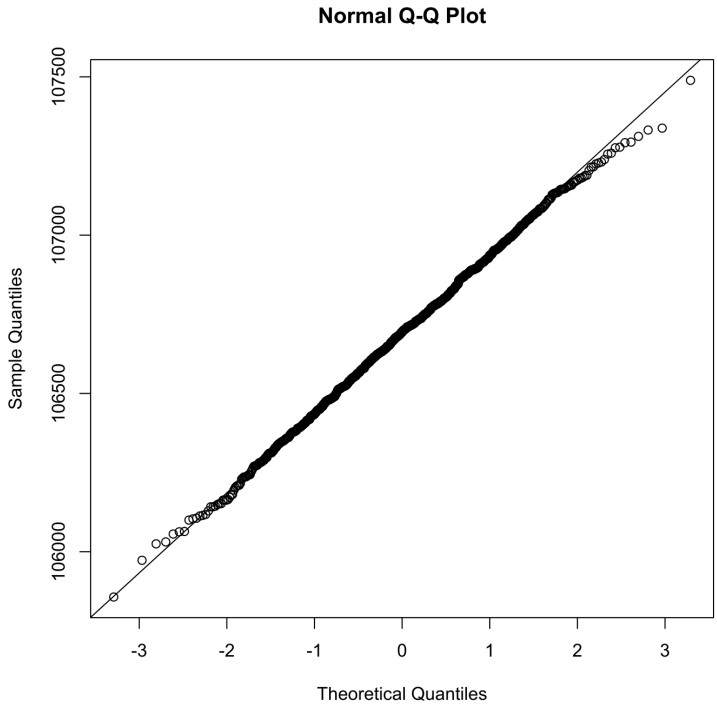
noise2.dat



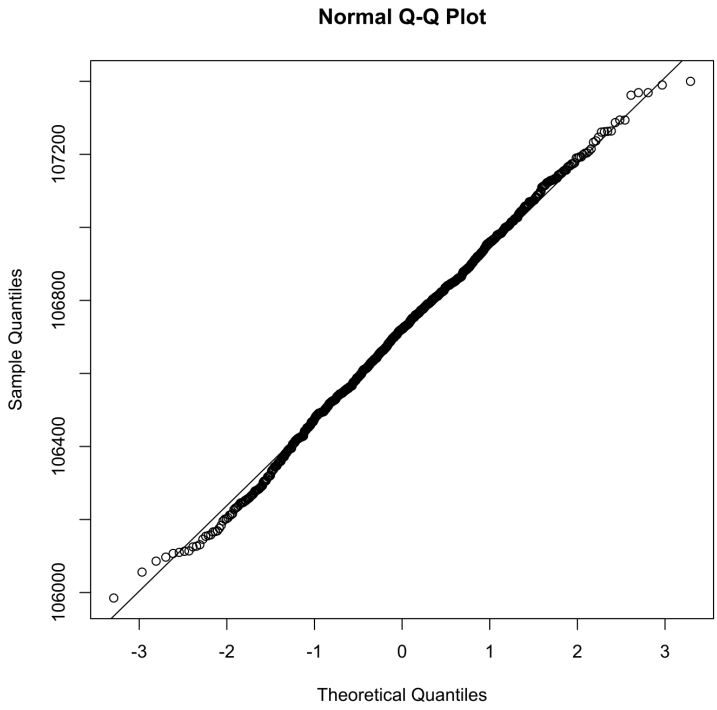



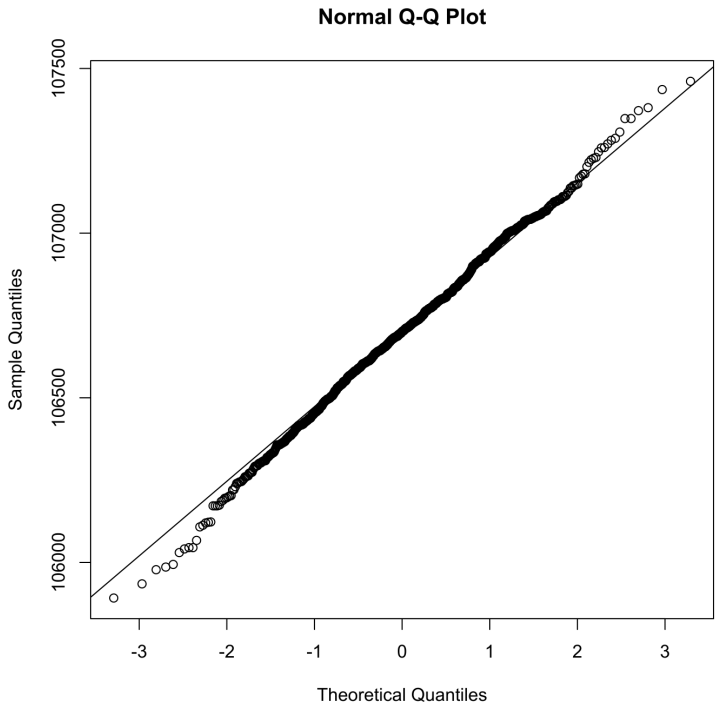
## QQ-plots [3]

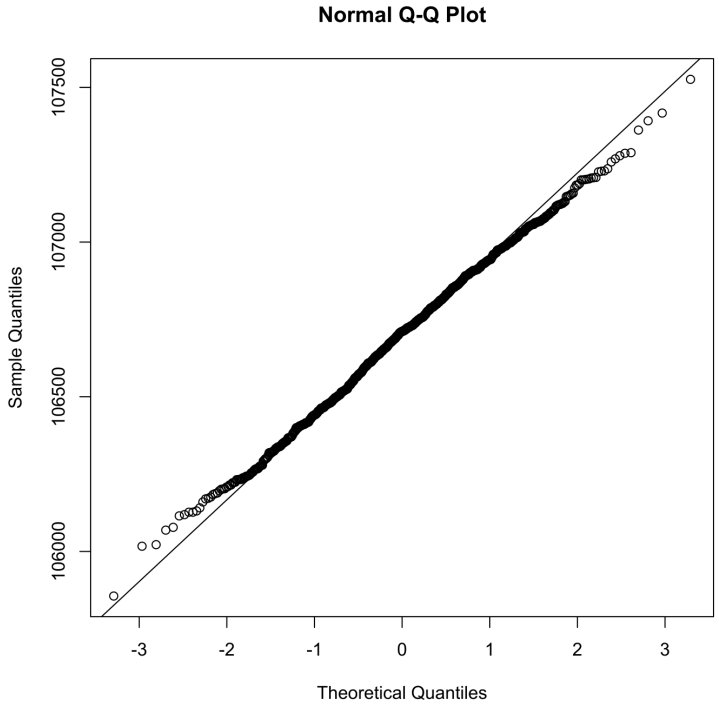


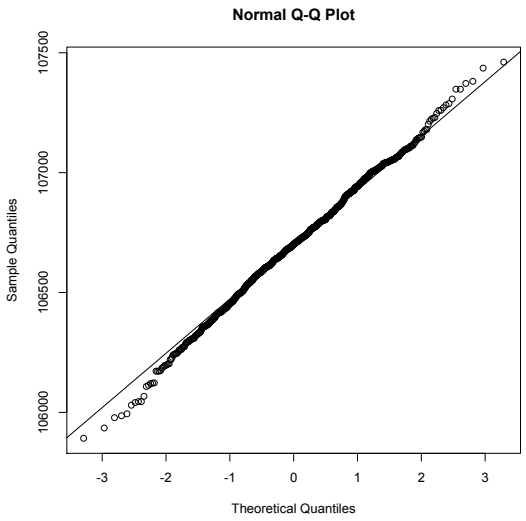




3 Duidelijkste manier om visueel na te gaan of data normaal verdeeld is of niet.
De volgorde van de plots is: lineair gegenereerd, 2 keer lineair met ruis, 2 keer random.

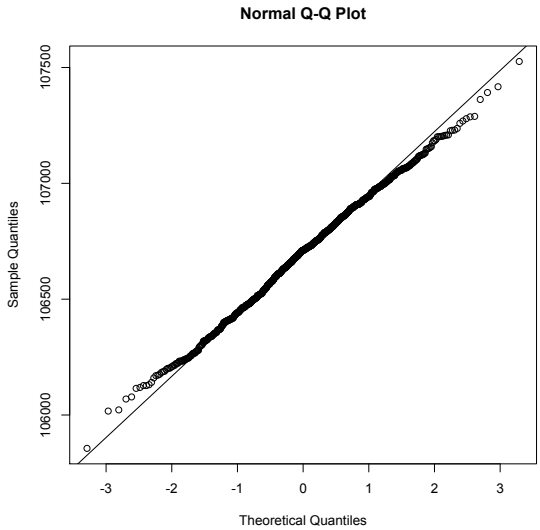

Normal Q-Q Plot
Sample Quantiles
106000
106400
106800
107200
-3
-2
-1
0
1
2
3
Theoretical Quantiles


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
106000
106500
107000
107500
-3
-2
-1
0
1
2
3


Normal Q-Q Plot
Sample Quantiles
Theoretical Quantiles
106000
106500
107000
107500
-3
-2
-1
0
1
2
3



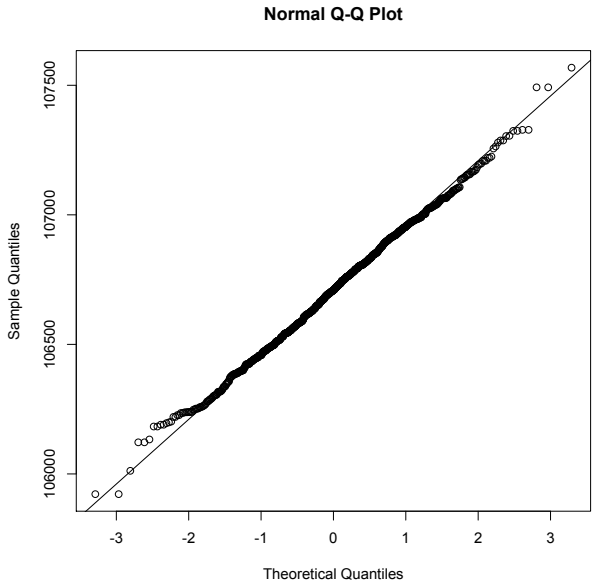

random1.dat




linear.dat

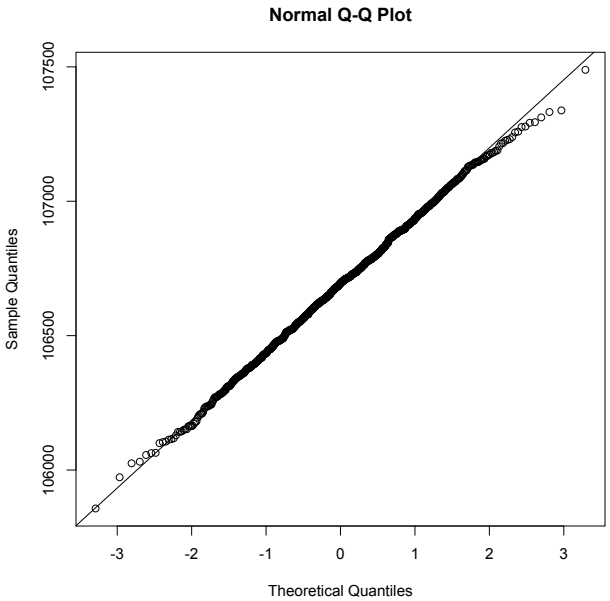


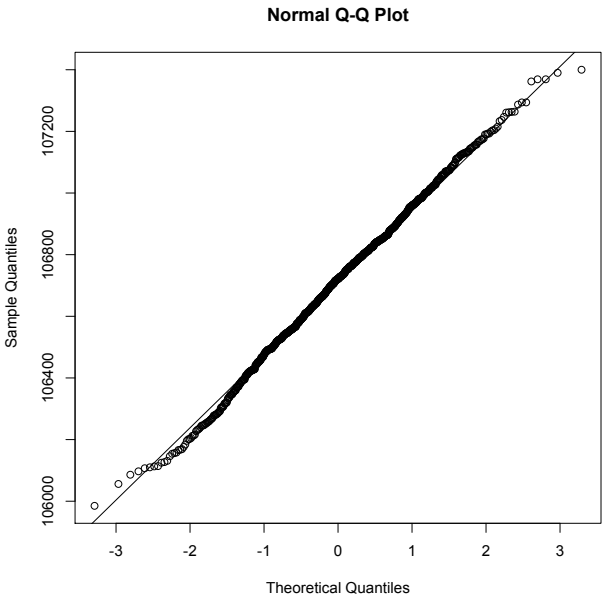

random2.dat



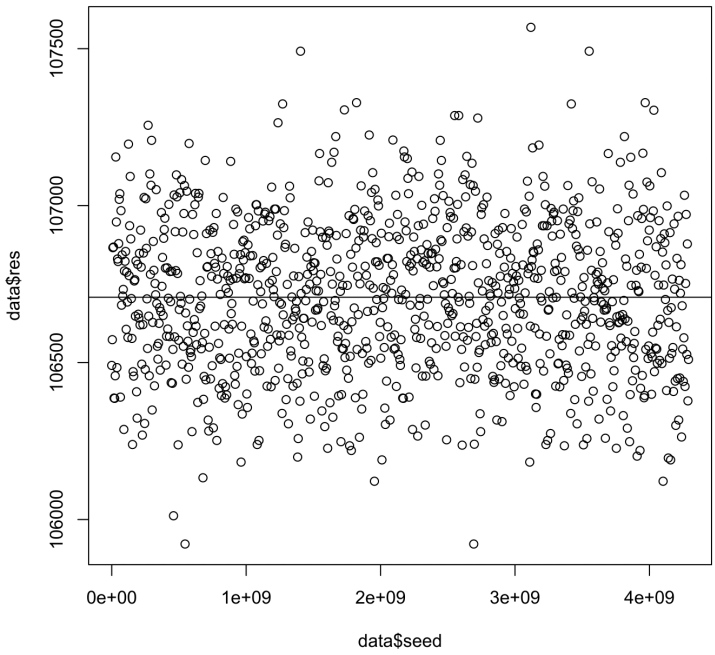

noise1.dat

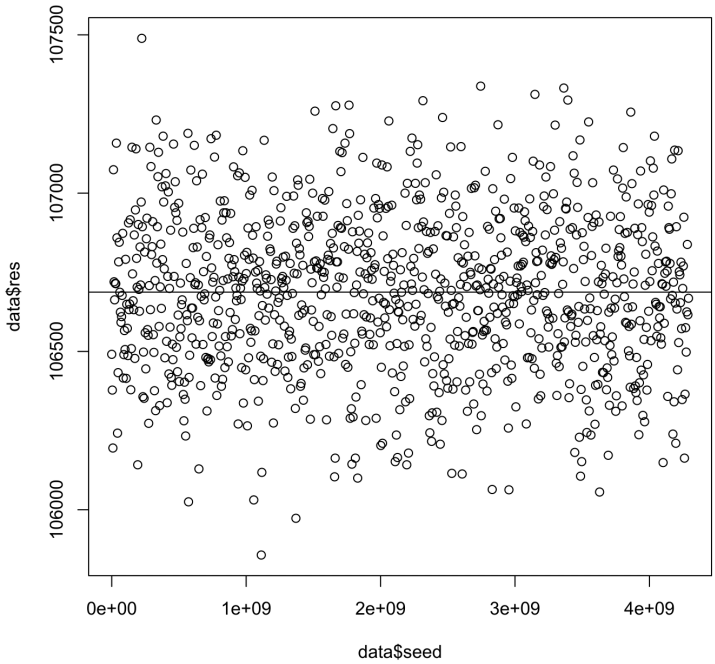
noise2.dat



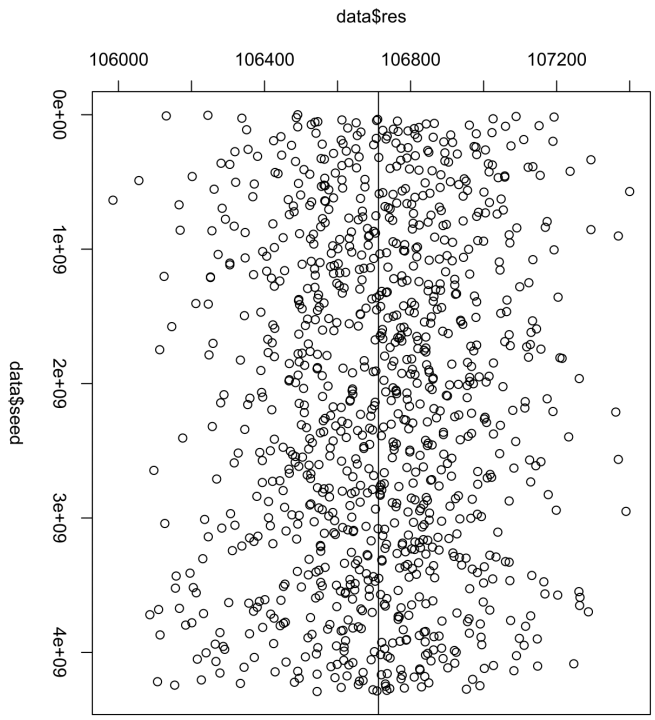



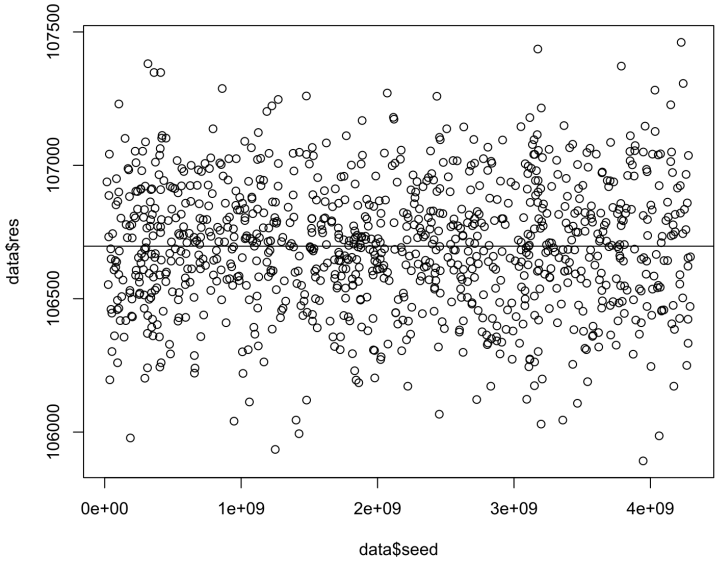
## Point plots [4]

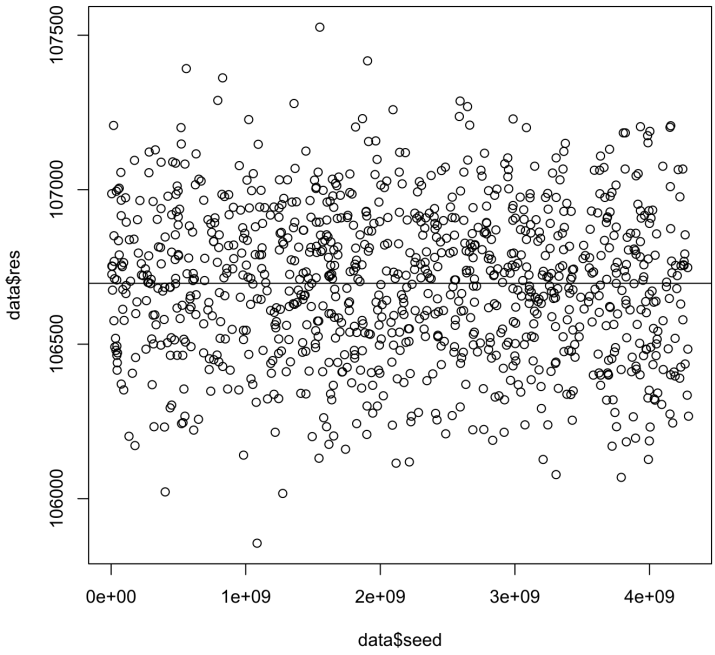


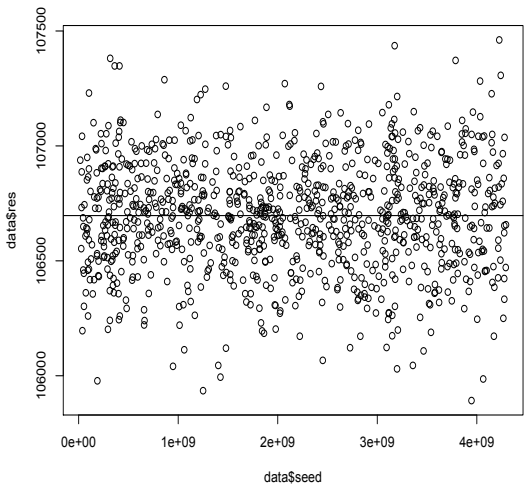




4 De volgorde van de plots is: lineair gegenereerd, 2 keer lineair met ruis, 2 keer random.

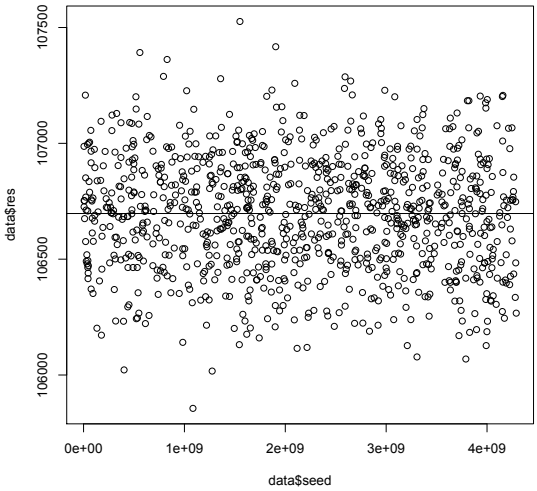

data$res
106000
106400
106800
107200
data$seed
0e+00
1e+09
2e+09
3e+09
4e+09


data$res
107500
107000
106500
106000
0e+00
1e+09
2e+09
3e+09
4e+09
data$seed


data$res
107500
107000
106500
106000
0e+00
1e+09
2e+09
3e+09
4e+09
data$seed



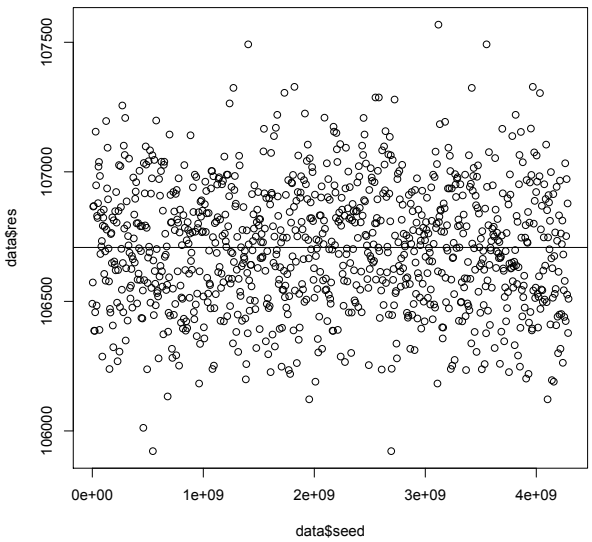

random1.dat




linear.dat




random2.dat




noise1.dat

noise2.dat