Análisis de datos

Se presenta todo lo que se hizo para llegar a los resultados. Muchas cosas son más de estadística que del proyecto, sin embargo, es mejor que no falte.

Preparación

Paquetes

```
library(readxl)
library(lme4)
```

```
## Loading required package: Matrix
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(qqplotr)
```

```
##
## Attaching package: 'qqplotr'

## The following objects are masked from 'package:ggplot2':
##
##     stat_qq_line, StatQqLine
```

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
library(lattice)
library(emmeans)
```

```
## Welcome to emmeans.
## Caution: You lose important information if you filter this package's results.
## See '? untidy'
```

Tengo entendido que para la tesis es necesario colocar los paquetes en algún lado, puede hacer la consulta.

Se cargan los datos. Es necesario pasar las observaciones a tipo factor.

```
base1=read_xlsx("final_registro_cloud.xlsx",col_names = FALSE)
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
```

```
names(base1)=c("Tiempo","BL","Nube","Denied","Right","Solicitudes","ID","Bloque")
base1$BL=as.factor(base1$BL)
base1$Solicitudes=as.factor(base1$Solicitudes)
base1$Nube=as.factor(base1$Nube)




base2=read_xlsx("final_registro_local.xlsx",col_names = FALSE)
```

```
## New names:
## * `` -> `...1`
## * `` -> `...2`
## * `` -> `...3`
## * `` -> `...4`
## * `` -> `...5`
## * `` -> `...6`
## * `` -> `...7`
## * `` -> `...8`
```

```
names(base2)=c("Tiempo","BL","Nube","Denied","Right","Solicitudes","ID","Bloque")
base2$BL=as.factor(base2$BL)
base2$Solicitudes=as.factor(base2$Solicitudes)
base2$Nube=as.factor(base2$Nube)
```

Podemos unir las bases para que sea solo una tabla grande y ver los tratamientos

```
base=rbind(base1,base2)
base$Bloque=as.factor(base$Bloque)
table(base$BL,base$Nube,base$Solicitudes)
```

```
## , ,  = 10
##
##
##      Nube Local
##   BL   90    80
##   ML   90    80
##
## , ,  = 100
##
##
##      Nube Local
##   BL  900   800
##   ML  895   800
##
## , ,  = 500
##
##
##      Nube Local
##   BL 3759  3866
##   ML 3758  3865
```

```
str(base)
```

```
## tibble [18,983 x 8] (S3: tbl_df/tbl/data.frame)
##  $ Tiempo     : num [1:18983] 0.0005 0.163288 0.000427 0.15167 0.000418 ...
##  $ BL         : Factor w/ 2 levels "BL","ML": 1 2 1 2 1 2 1 2 1 2 ...
##  $ Nube       : Factor w/ 2 levels "Nube","Local": 1 1 1 1 1 1 1 1 1 1 ...
##  $ Denied     : chr [1:18983] "Denied" "Aproved" "Aproved" "Aproved" ...
##  $ Right      : chr [1:18983] "Right" "Right" "Right" "Right" ...
##  $ Solicitudes: Factor w/ 3 levels "10","100","500": 1 1 1 1 1 1 1 1 1 1 ...
##  $ ID         : num [1:18983] 1 2 3 4 5 6 7 8 9 10 ...
##  $ Bloque     : Factor w/ 51 levels "L1001","L1002",..: 34 34 34 34 34 34 34 34 34 34 ...
```

Vemos que hay un desbalance con los tratamientos, según se pueden eliminar parcelas, se eliminan las que causan el desbalance.

```
quitar1=which(base1$Bloque=="N109")
base1=base1[-quitar1,]

quitar2=which(base1$Bloque=="N1009")
base1=base1[-quitar2,]
```

```
quitar3=which(base1$Bloque=="N5009")
base1=base1[-quitar3,]

quitar4=which(base1$Bloque=="N5006")
base1=base1[-quitar4,]

quitar5=which(base2$Bloque=="L5006")
base2=base2[-quitar5,]

quitar6=which(base1$Bloque=="N5008")
base1=base1[-quitar6,]

quitar7=which(base2$Bloque=="L5008")
base2=base2[-quitar7,]

quitar8=which(base1$Bloque=="N5004")
base1=base1[-quitar8,]

quitar9=which(base2$Bloque=="L5004")
base2=base2[-quitar9,]

quitar10=which(base1$Bloque=="N1007")
base1=base1[-quitar10,]

quitar11=which(base2$Bloque=="L1007")
base2=base2[-quitar11,]

base2$Bloque=base1$Bloque
base=rbind(base1,base2)
table(base$BL,base$Nube,base$Solicitudes)
```

```
## , ,  = 10
##
##
##      Nube Local
##   BL   80    80
##   ML   80    80
##
## , ,  = 100
##
##
##      Nube Local
##   BL  700   700
##   ML  700   700
##
## , ,  = 500
##
##
##      Nube Local
##   BL 2500  2500
##   ML 2500  2500
```

Se soluciona el desbalance, ahora los bloques deben de coincidir.

Estadística Descriptiva.

Promedios por tratamiento y varianza.

```
options(scipen = 999)
m=tapply(base$Tiempo,list(base$BL,base$Nube,base$Solicitudes),mean);m
```

```
## , , 10
##
##         Nube       Local
## BL 0.0008677 0.002882075
## ML 0.1918961 0.898266438
##
## , , 100
##
##            Nube       Local
## BL 0.0008544457 0.002883869
## ML 0.1564820486 0.907232110
##
## , , 500
##
##            Nube      Local
## BL 0.0009379336 0.00289288
## ML 0.1561837404 0.89883775
```

```
v=tapply(base$Tiempo,list(base$BL,base$Nube,base$Solicitudes),var);v
```

```
## , , 10
##
##                Nube          Local
## BL 0.0000002244848 0.000008591479
## ML 0.0315259459741 0.002696818658
##
## , , 100
##
##                Nube          Local
## BL 0.0000004528595 0.000009014041
## ML 0.0000608513538 0.007039751061
##
## , , 500
##
##               Nube          Local
## BL 0.000004491367 0.000009103962
## ML 0.000038641015 0.005070112955
```

Observamos que son valores que están cercanos a 0, esto se debe a que la variable respuesta está dada en segundos y pero los valores de tiempo son mucho más pequeños que un segundo.

```
base$TiempoT=base$Tiempo*100
m=tapply(base$TiempoT,list(base$BL,base$Nube,base$Solicitudes),mean);m
```

```
## , , 10
##
##        Nube      Local
## BL  0.08677  0.2882075
## ML 19.18961 89.8266438
##
## , , 100
##
##           Nube      Local
## BL  0.08544457  0.2883869
## ML 15.64820486 90.7232110
##
## , , 500
##
##           Nube     Local
## BL  0.09379336  0.289288
## ML 15.61837404 89.883775
```

```
v=tapply(base$TiempoT,list(base$BL,base$Nube,base$Solicitudes),var);v
```

```
## , , 10
##
##             Nube       Local
## BL   0.002244848  0.08591479
## ML 315.259459741 26.96818658
##
## , , 100
##
##           Nube       Local
## BL 0.004528595  0.09014041
## ML 0.608513538 70.39751061
##
## , , 500
##
##          Nube       Local
## BL 0.04491367  0.09103962
## ML 0.38641015 50.70112955
```

Parece que en *centisegundos* todo tiene un poco más de coherencia.

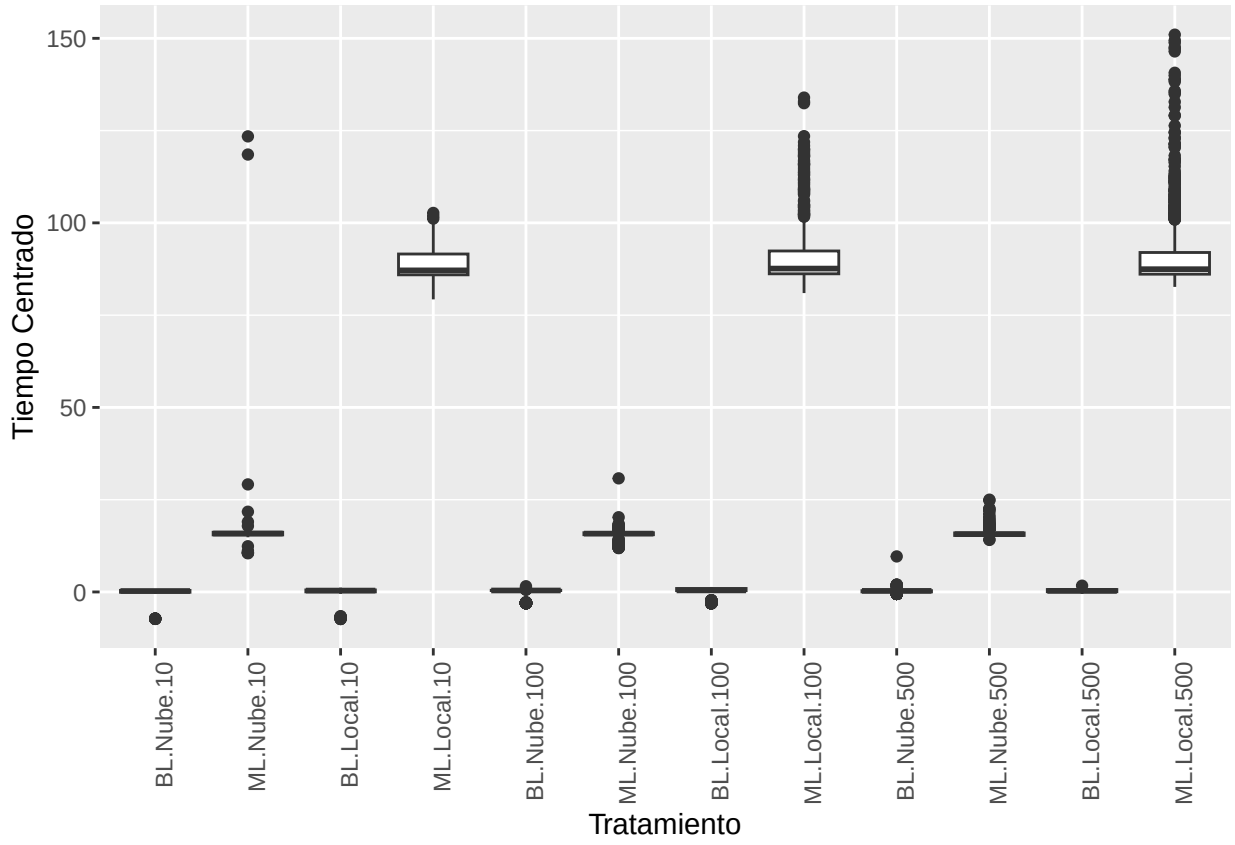
Análisis gráfico

Gráfico de cajas

Este puede ayudarnos a comprender los tratamientos, como se comparan entre ellos en términos de medias y varianzas. Como se trata de un modelo con bloques hay que centras los datos.

```
modc=lm(TiempoT~Bloque,data = base)
fit=modc$fit
rc=base$TiempoT-fit+mean(base$TiempoT)
base$rc=rc
```

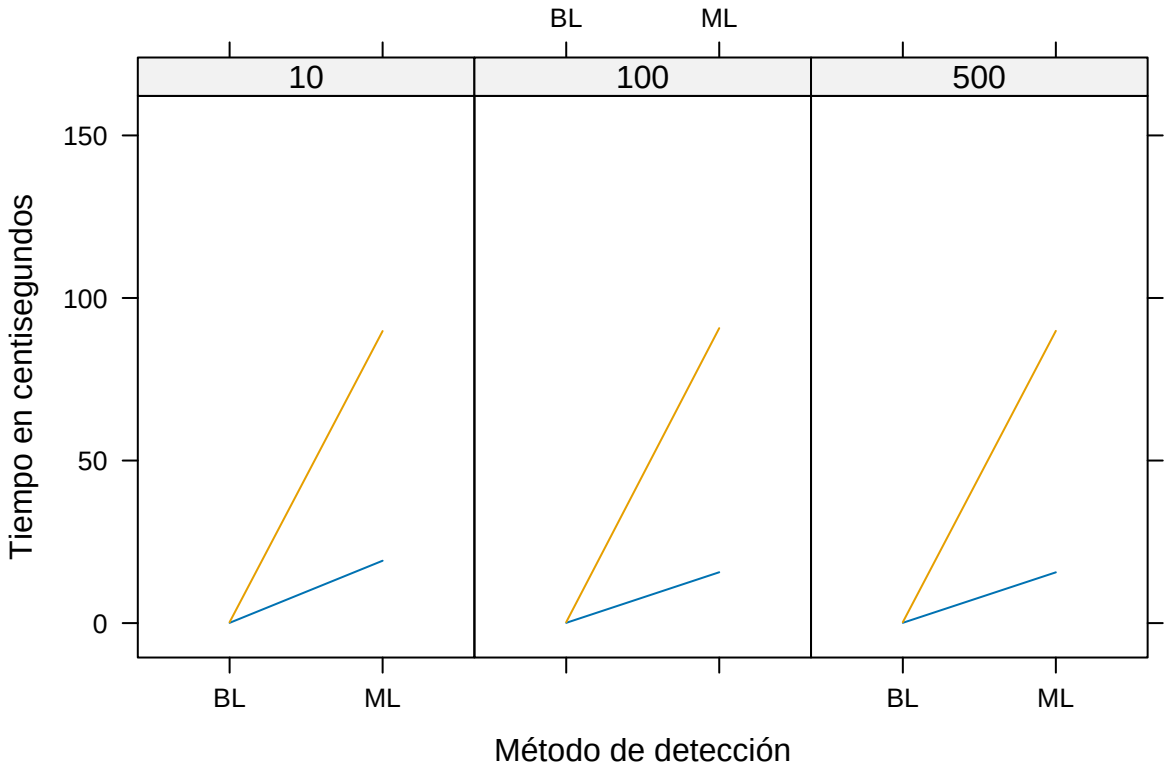
```
ggplot(base, aes(x = interaction(BL, Nube, Solicitudes), y = rc)) +
  geom_boxplot() +
  labs(x = "Tratamiento", y = "Tiempo Centrado") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```



Observamos un gráfico de cajas, la lineas negras representan el promedio de cada tratamiento (cada combinación de cada nivel de cada factor)

Gráfico de lineas para ver Interacción Triple.

```
xyplot(base$TiempoT~base$BL|base$Solicitudes,group=base$Nube,pch=18,type="a",xlab="Método de detección"
```

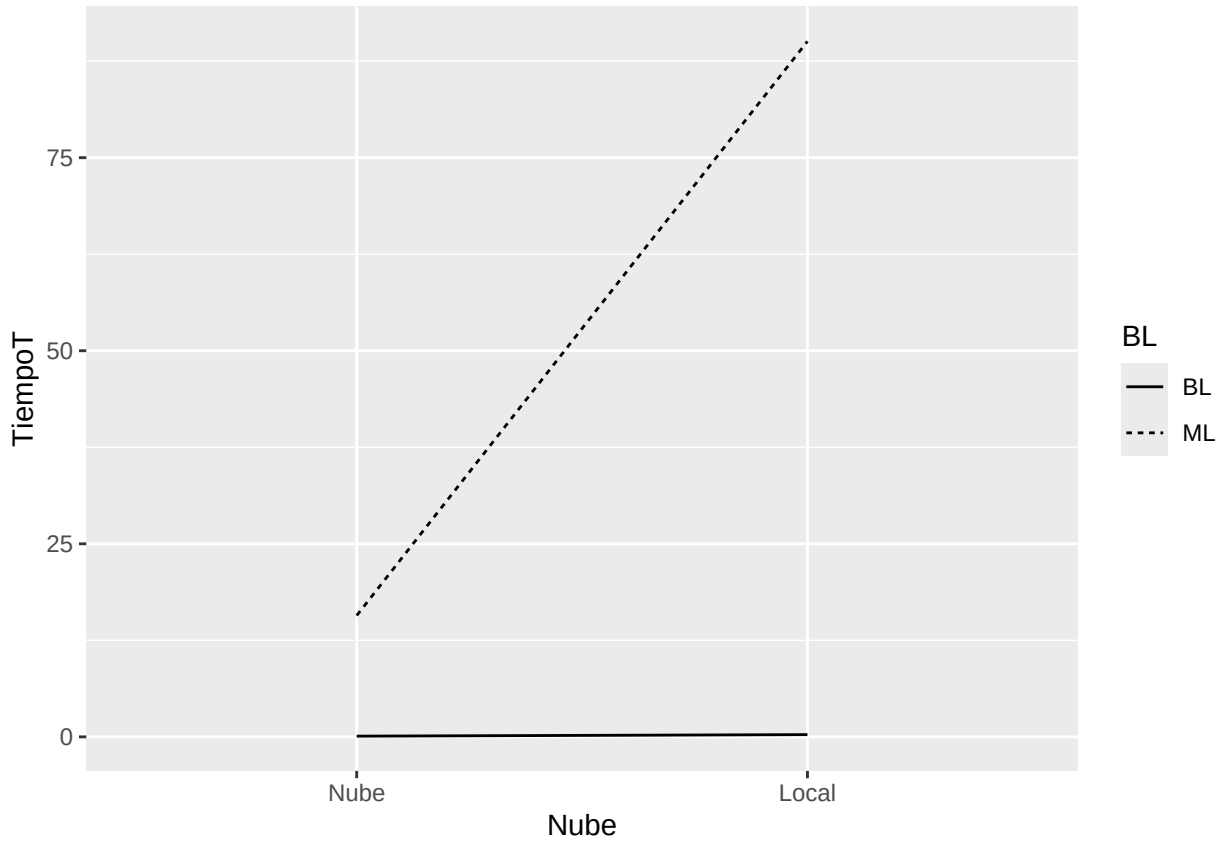


Gráficamente no se observa interacciónm triple, siemore es necesaria una prueba estadística para confirmarlo pero es un buen indicio.

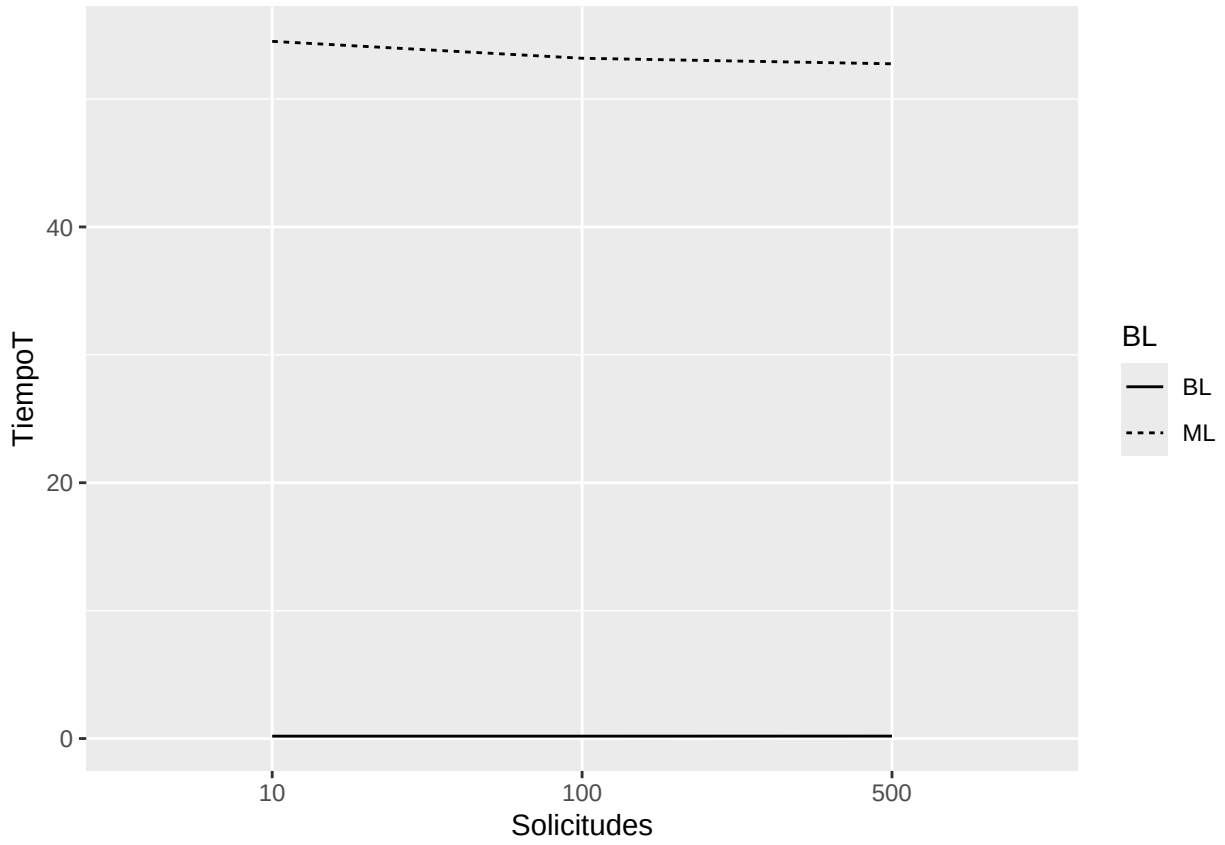
Gráficos de lineas para ver interacción doble.

```
ggplot(base,aes(x=Nube,y=TiempoT,group = BL))+
  stat_summary(fun.y = "mean",geom = "line",aes(linetype = BL))
```

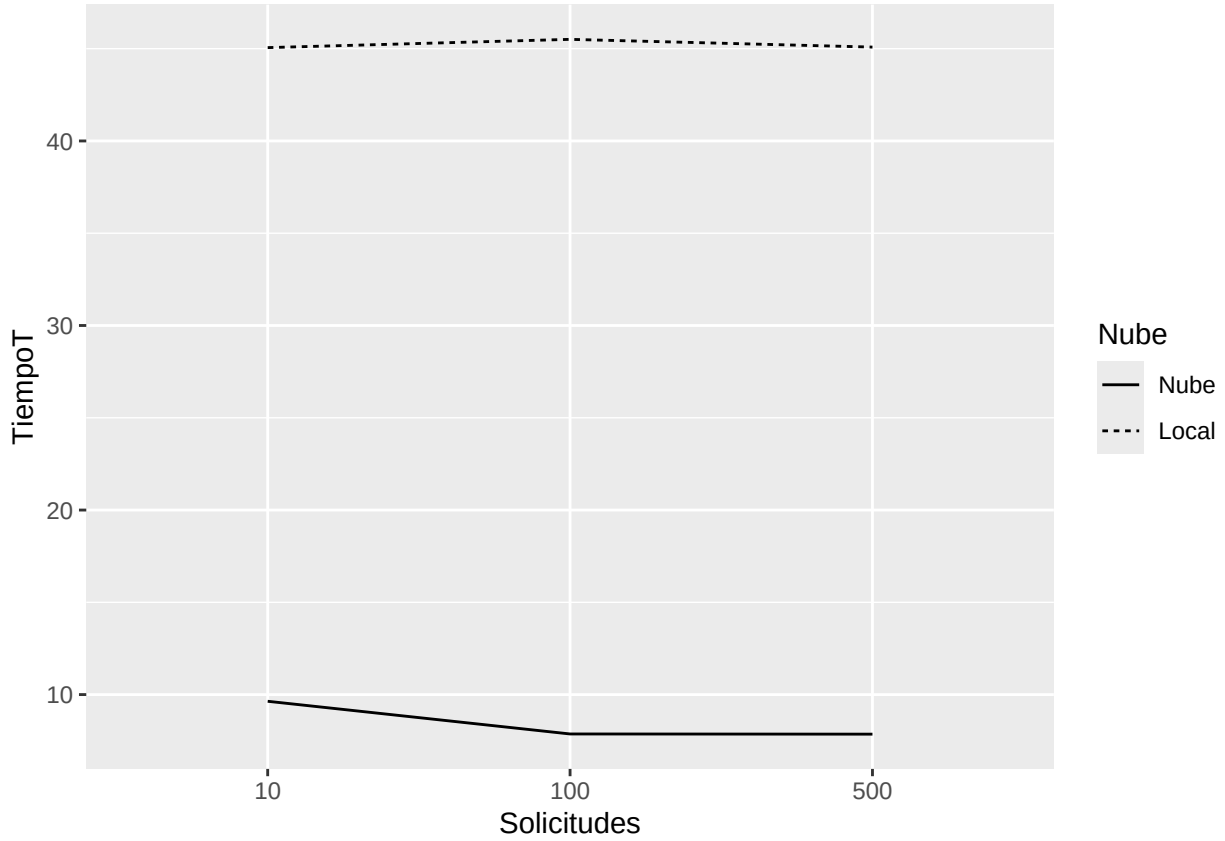
```
## Warning: The ‘fun.y‘ argument of ‘stat_summary()‘ is deprecated as of ggplot2 3.3.0.
## i Please use the ‘fun‘ argument instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```




```
ggplot(base,aes(x=Solicitudes,y=TiempoT,group = BL))+
  stat_summary(fun.y = "mean",geom = "line",aes(linetype = BL))
```




```
ggplot(base,aes(x=Solicitudes,y=TiempoT,group = Nube))+
  stat_summary(fun.y = "mean",geom = "line",aes(linetype = Nube))
```



Solo se observa interacción entre Ubicación y Método de detección, vemos que las diferencias entre Black List y Machine Learning son más pequeñas en la Nube que en Local.

El modelo

El modelo es un modelo mixto de parcelas divididas, el cual debe de cumplir los supuestos de normalidad y de homocedasticidad. Este no se trata del modelo completo (Adjunto más adelante)

Es el siguiente:

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + (\alpha\delta)_{ik} + (\beta\delta)_{jk} + (\alpha\beta\delta)_{ijk} + \nu_l$$

Donde:

- μ es la media general
- α_i es el efecto del nivel i del Factor BlackList
- β_j es el efecto del nivel j del Factor Nube
- δ_k es el efecto del nivel k del Factor Solicitudes
- $(\alpha\beta)_{ij}$ es el efecto de la interacción entre BlackList y Nube
- $(\alpha\delta)_{ik}$ es el efecto de la interacción entre BlackList y Solicitudes
- $(\beta\delta)_{jk}$ es el efecto de la interacción entre Nube y Solicitudes
- ν_l Es el efecto de la parcela.

Se ajusta un modelo mixto

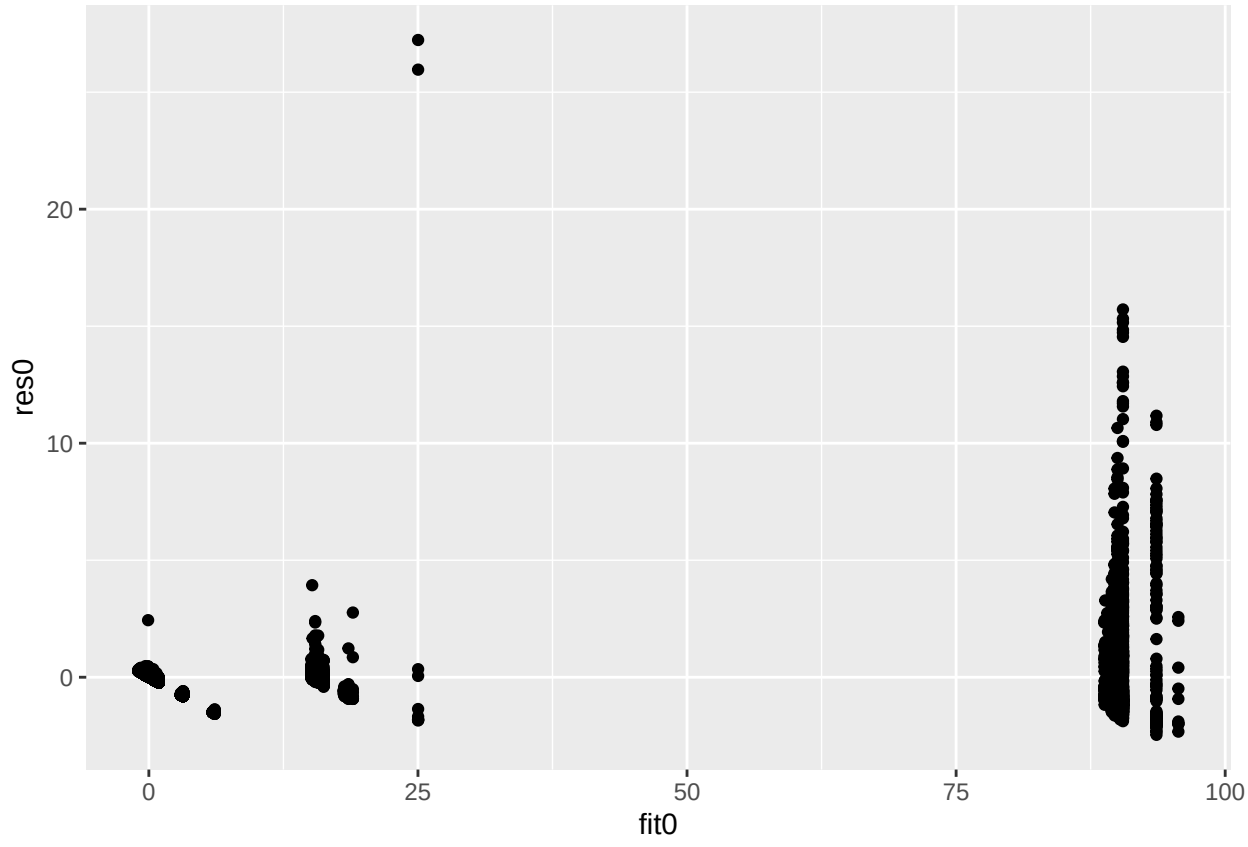
```
modp0=lmer(TiempoT~BL*Nube*Solicitudes+(1|Bloque),data = base)
summ0=summary(modp0)
```

Supuestos del modelo

A partir de acá es pura estadística. Lo importante es comprender lo que hablamos, los modelos asumen ciertos elementos sobre los datos y estos datos, de la forma en los que los teníamos no cumplían ningún supuesto.

Se asume el supuesto de independencia, se analiza entonces homocedasticidad y normalidad. Se oueden usar qqPlots, la prueba de Kolmogorov-Smirnov, el gráfico de predichos contra residuos y la prueba de levene.

```
res0=summ0$residuals
fit0=fitted(modp0)
dffp=data.frame(res0,fit0)


ggplot(data = dffp, aes(x = fit0, y = res0)) +
  geom_point()
```




```
leveneTest(res0~BL*Nube*Solicitudes,data = base)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value                Pr(>F)
```

```
## group    11   243.1 < 0.00000000000000022 ***
##       13108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tanto en el gráfico de residuos contra predichos como en la prueba de Levene, no hay razón para creer que hay homocedasticidad.

```
ks.test(res0,"pnorm")
```

```
## Warning in ks.test.default(res0, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  res0
## D = 0.30686, p-value < 0.00000000000000022
## alternative hypothesis: two-sided
```

```
residuos=as.data.frame(res0)
res.stnd = scale(residuos$res0) # <- Residuos estandarizados.
ggplot(data = residuos, mapping = aes(sample = res.stnd )) + stat_qq_point() +
      ylab("Residuos Estandarizados") + xlab("Norm Quantiles") + ggtitle("Normal QQ Plot")  +
```

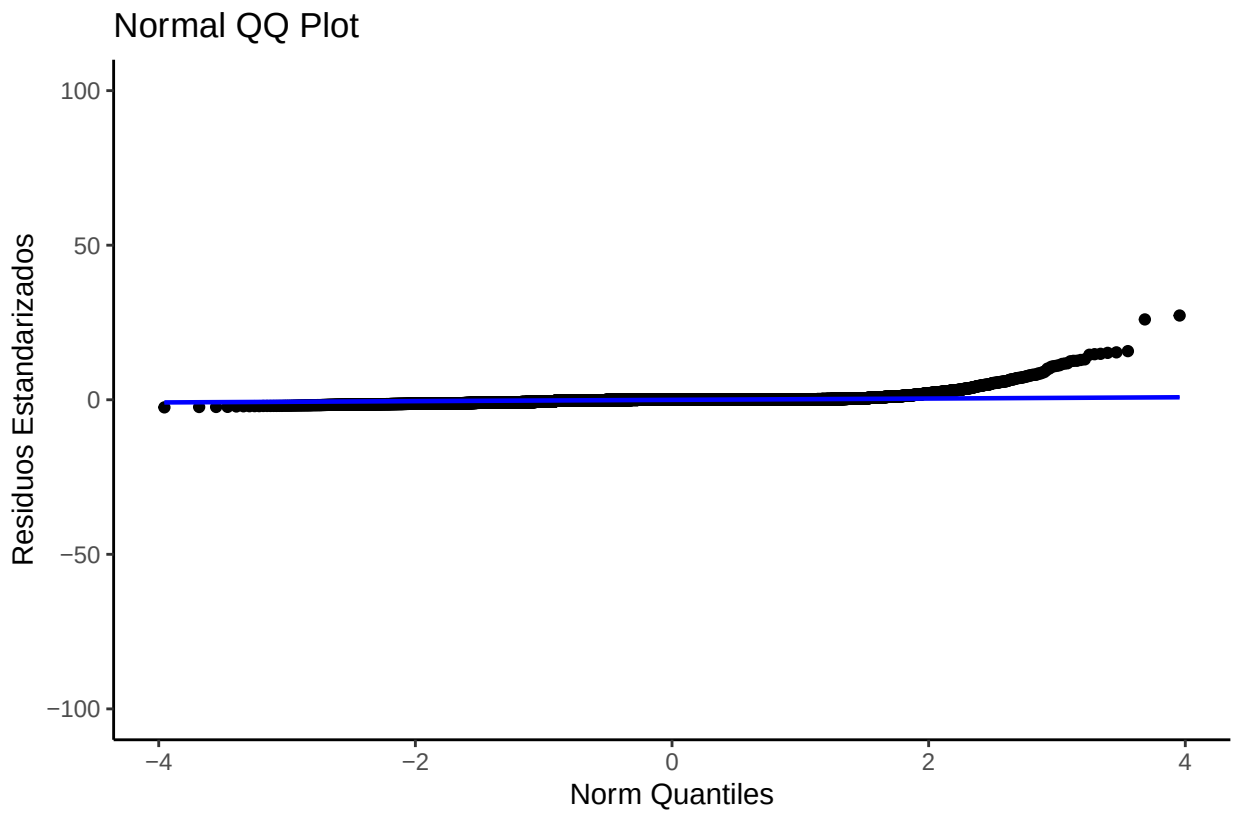


Grafico 1

De igual forma, tanto en el gráfico como en la prueba de Kolmogorov se puede distinguir normalidad.

En ambos gráficos hay un dato extremo.

Vamos a intentar cambiar la respuesta por el promedio del tratamiento al que pertenecen

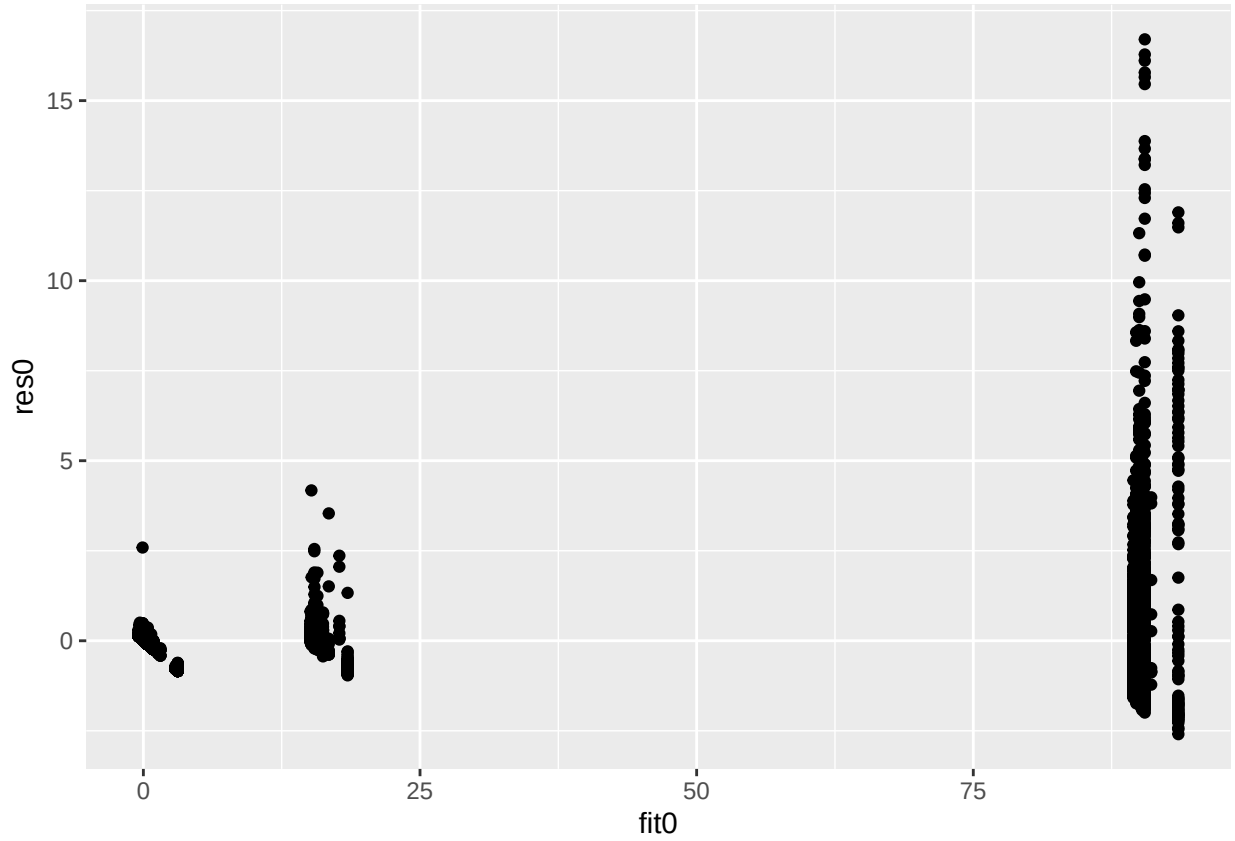
```
which(res0>20,)
```

```
## 5325 5327
## 5325 5327
```

```
#tapply(base$TiempoT,list(base$BL,base$Nube,base$Solicitudes),mean) ##El promedio es 19 y estos tienen
baseA=base
baseA[c(5325,5327),9]=c(19.18961,19.18961)
```

Podemos volver a hacer todo a ver qué tal nos va

```
modp0=lmer(TiempoT~BL*Nube*Solicitudes+(1|Bloque),data = baseA)
summ0=summary(modp0)

# Homocedasticidad

res0=summ0$residuals
fit0=fitted(modp0)
dffp=data.frame(res0,fit0)


ggplot(data = dffp, aes(x = fit0, y = res0)) +
  geom_point()
```




```
leveneTest(res0~BL*Nube*Solicitudes,data = baseA)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value                Pr(>F)
## group    11  279.29 < 0.00000000000000022 ***
##       13108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Normalidad

ks.test(res0,"pnorm")
```

```
## Warning in ks.test.default(res0, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  res0
## D = 0.30599, p-value < 0.00000000000000022
## alternative hypothesis: two-sided
```

```
residuos=as.data.frame(res0)
res.stnd = scale(residuos$res0) # <- Residuos estandarizados.
ggplot(data = residuos, mapping = aes(sample = res.stnd )) + stat_qq_point() +
      ylab("Residuos Estandarizados") + xlab("Norm Quantiles") + ggtitle("Normal QQ Plot")  +
```

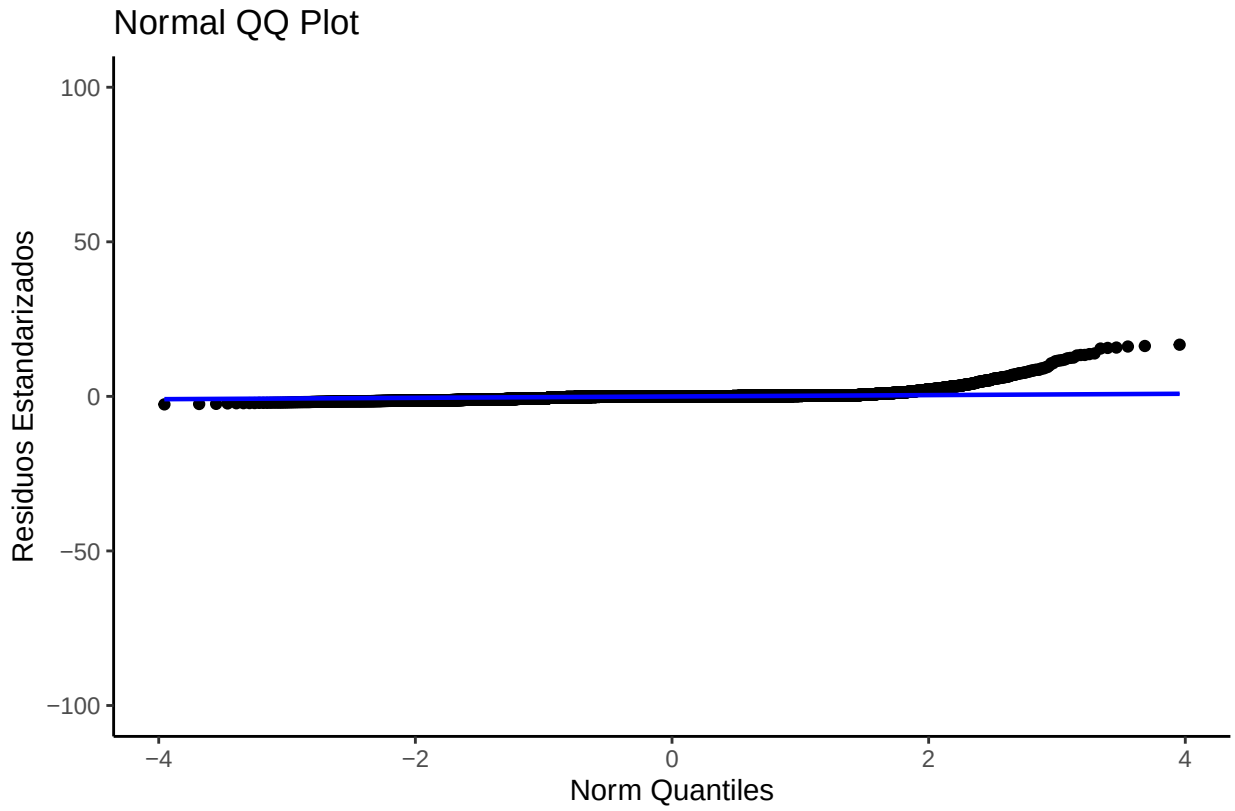


Grafico 1

No mejoró mucho

Por esta razón vamos a intentar a justar un modelo Generalizado mixto, a ver si eso nos ayuda a arreglar los supuestos.

Modelo generalizado Mixto

De igual forma se probó con un modelo más complejo y no se obtuvo ningún resultado distinto.

```
modp1=glmer(TiempoT~BL*Nube*Solicitudes+(1|Bloque),family = Gamma(link = "log"),data = base)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.00756635 (tol = 0.002, component 1)
```

```
summ=summary(modp1)
summ
```

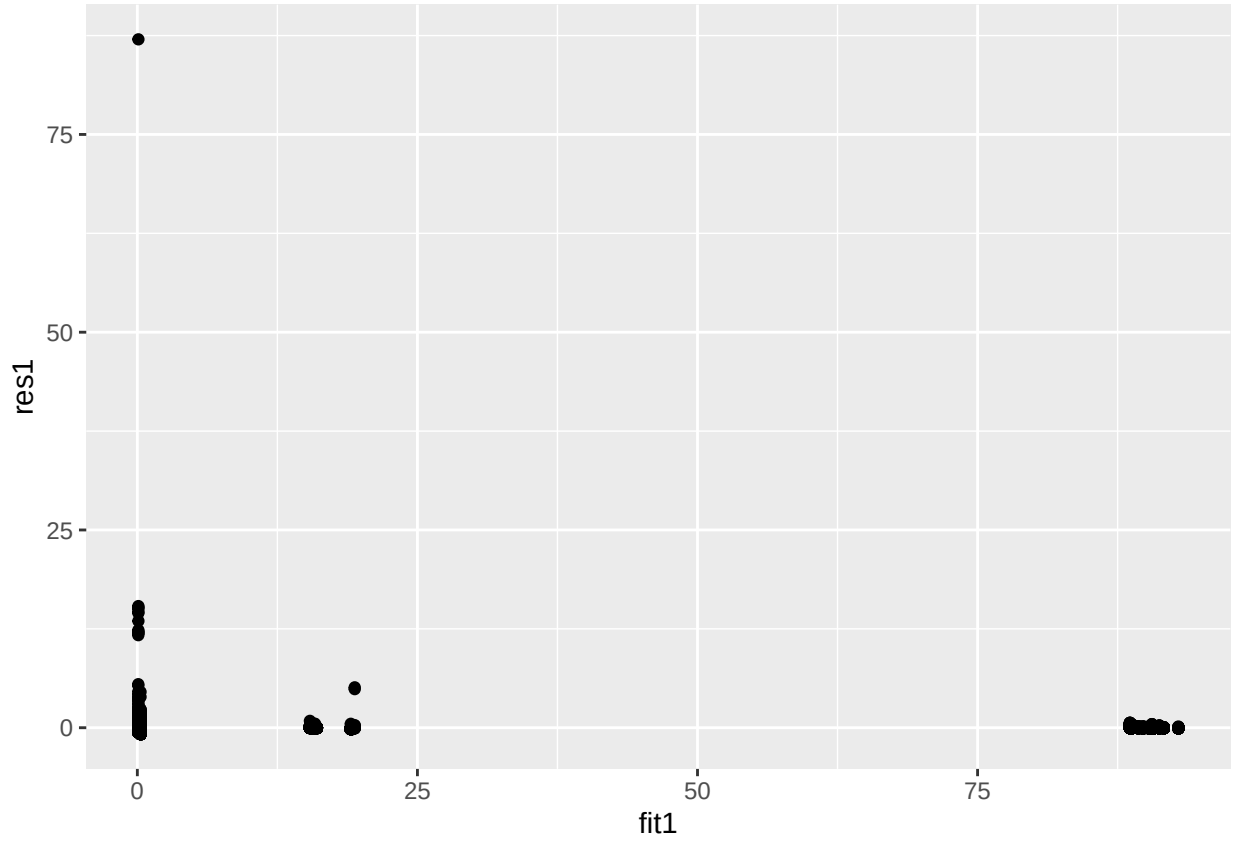
```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
```

```
##  Family: Gamma  ( log )
## Formula: TiempoT ~ BL * Nube * Solicitudes + (1 | Bloque)
##    Data: base
##
##      AIC      BIC   logLik deviance df.resid
##  46062.5  46167.2 -23017.2  46034.5    13106
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -0.813 -0.221 -0.020  0.034 87.023
##
## Random effects:
##  Groups   Name        Variance Std.Dev.
##  Bloque   (Intercept) 0.001198 0.03462
##  Residual             1.264358 1.12444
## Number of obs: 13120, groups:  Bloque, 20
##
## Fixed effects:
##                               Estimate Std. Error t value             Pr(>|z|)
## (Intercept)                  -2.44514    0.07475 -32.713 < 0.0000000000000002
## BLML                          5.39576    0.10500  51.388 < 0.0000000000000002
## NubeLocal                     1.19966    0.10492  11.434 < 0.0000000000000002
## Solicitudes100               -0.01651    0.07947  -0.208              0.83544
## Solicitudes500                0.07735    0.07694   1.005              0.31478
## BLML:NubeLocal                0.34751    0.14847   2.341              0.01925
## BLML:Solicitudes100          -0.18366    0.11087  -1.657              0.09761
## BLML:Solicitudes500          -0.27948    0.10667  -2.620              0.00879
## NubeLocal:Solicitudes100      0.01742    0.11076   0.157              0.87503
## NubeLocal:Solicitudes500     -0.07201    0.10659  -0.676              0.49932
## BLML:NubeLocal:Solicitudes100  0.19293    0.15672   1.231              0.21831
## BLML:NubeLocal:Solicitudes500  0.27514    0.15083   1.824              0.06812
##
## (Intercept)                  ***
## BLML                         ***
## NubeLocal                    ***
## Solicitudes100
## Solicitudes500
## BLML:NubeLocal               *
## BLML:Solicitudes100          .
## BLML:Solicitudes500          **
## NubeLocal:Solicitudes100
## NubeLocal:Solicitudes500
## BLML:NubeLocal:Solicitudes100
## BLML:NubeLocal:Solicitudes500 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) BLML   NubLcl Slc100 Slc500 BLML:NbL BLML:S1 BLML:S5 NL:S10
## BLML        -0.701
## NubeLocal   -0.702  0.500
## Solictds100 -0.940  0.660  0.660
## Solictds500 -0.971  0.681  0.682  0.914
## BLML:NubLcl  0.496 -0.708 -0.707 -0.467 -0.482
```

```
## BLML:Slc100  0.664 -0.947 -0.474 -0.696 -0.645  0.671
## BLML:Slc500  0.690 -0.984 -0.492 -0.649 -0.692  0.697    0.933
## NbLcl:Sl100  0.665 -0.474 -0.947 -0.697 -0.646  0.670    0.500   0.467
## NbLcl:Sl500  0.691 -0.492 -0.984 -0.650 -0.693  0.696    0.467   0.500   0.933
## BLML:NL:S10 -0.470  0.670  0.670  0.493  0.456 -0.947   -0.708  -0.660  -0.707
## BLML:NL:S50 -0.488  0.697  0.696  0.459  0.489 -0.984   -0.660  -0.708  -0.659
##             NL:S50 BLML:NL:S1
## BLML
## NubeLocal
## Solictds100
## Solictds500
## BLML:NubLcl
## BLML:Slc100
## BLML:Slc500
## NbLcl:Sl100
## NbLcl:Sl500
## BLML:NL:S10 -0.659
## BLML:NL:S50 -0.707  0.933
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.00756635 (tol = 0.002, component 1)
```

Supuestos

```
res1=summ$residuals
fit1=fitted(modp1)
dffp1=data.frame(res1,fit1)


ggplot(data = dffp, aes(x = fit1, y = res1)) +
  geom_point()
```




```
leveneTest(res1~BL*Nube*Solicitudes,data = base)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##          Df F value                Pr(>F)
## group    11  115.79 < 0.00000000000000022 ***
##       13108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
ks.test(res1,"pnorm")
```

```
## Warning in ks.test.default(res1, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  res1
## D = 0.27491, p-value < 0.00000000000000022
## alternative hypothesis: two-sided
```

```
residuos1=as.data.frame(res1)
res.stnd1 = scale(residuos1$res1) # <- Residuos estandarizados.
ggplot(data = residuos1, mapping = aes(sample = res.stnd1 )) + stat_qq_point() +
      ylab("Residuos Estandarizados") + xlab("Norm Quantiles") + ggtitle("Normal QQ Plot")  +
```

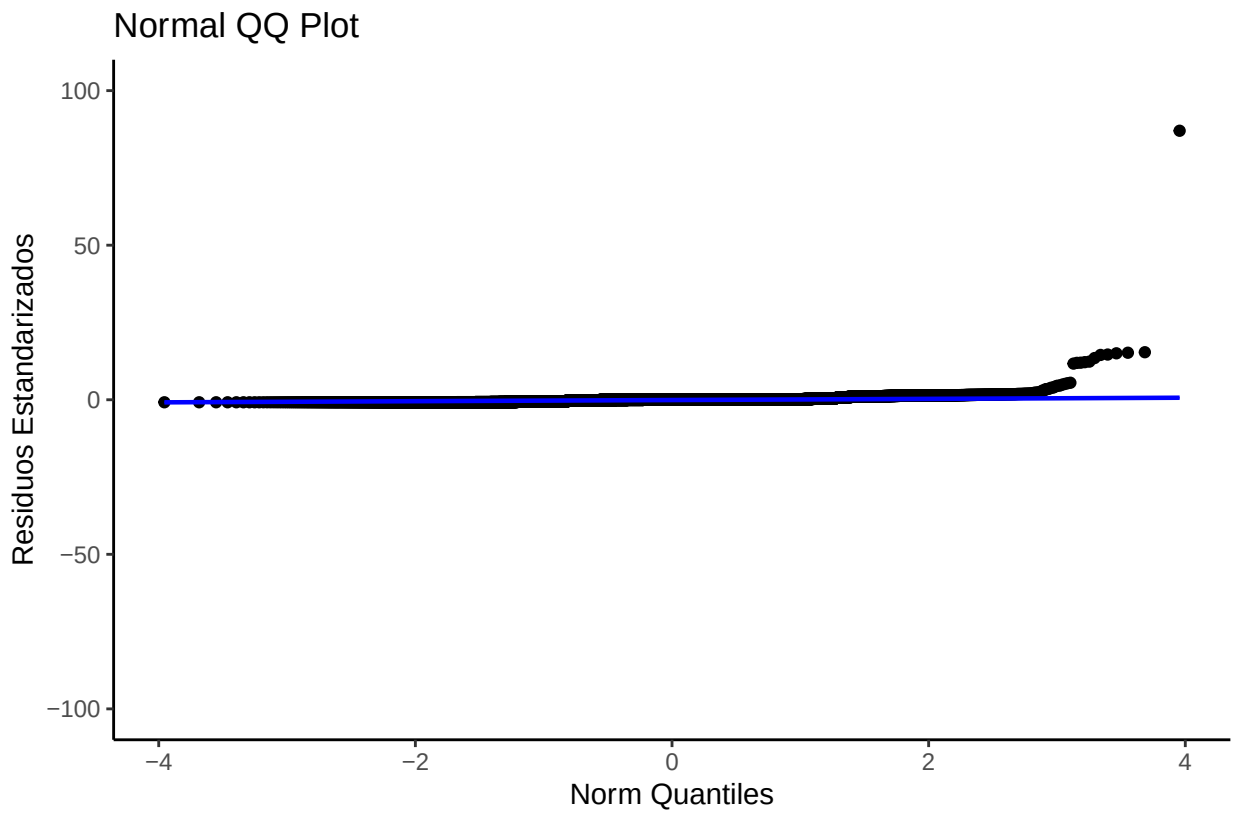


Grafico 1

Observamos que la normalidad se arregla un poco, pero la homocedasticidad sigue dando problemas. Puede que los valores extemos sigan afectando.

En este caso hay un punto en especifico que causa problemas, se va a cambiar por el promedio del tratamiento

```
which(res1>75);which(res.stnd1>75) ## Se trata del mismo
```

```
## 1465
## 1465
```

```
## [1] 1465
```

```
base[1465,]
```

```
## # A tibble: 1 x 10
##   Tiempo BL    Nube  Denied Right Solicitudes    ID Bloque TiempoT    rc
##    <dbl> <fct> <fct> <chr>  <chr> <fct>       <dbl> <chr>    <dbl> <dbl>
## 1 0.0940 BL    Nube  Denied Right 500          1465 N5002     9.40  9.63
```

```
0.09379336  ## Es el promedio del tratamiento que es BL,Nube y 500 y Tarda 9!
```

```
## [1] 0.09379336
```

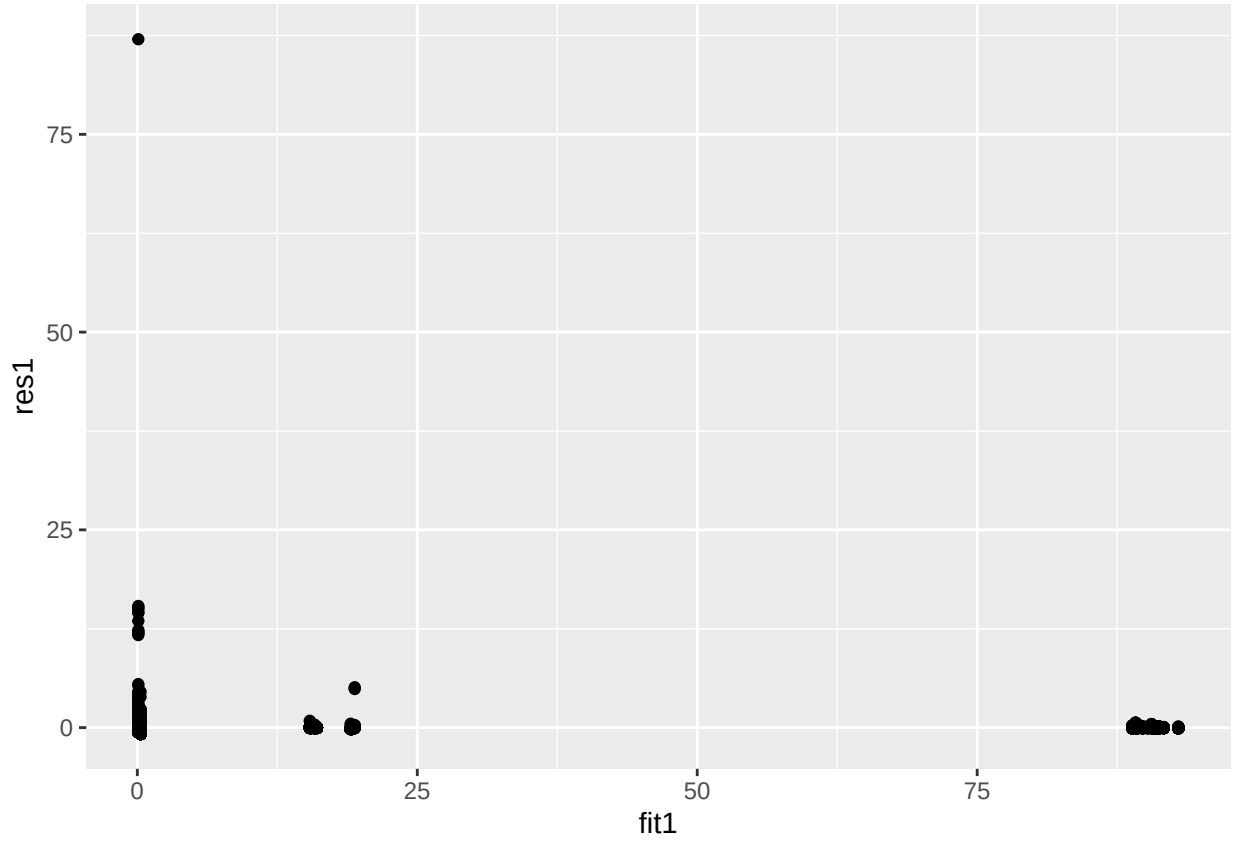
```
baseB=base
baseB[1465,9]=0.09379336
```

Y podemos volver a hacer todo a ver como nos da.

```
modp1=glmer(TiempoT~BL*Nube*Solicitudes+(1|Bloque),family = Gamma(link = "log"),data = baseB)
```

```
## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, :
## Model failed to converge with max|grad| = 0.0180776 (tol = 0.002, component 1)
```

```
# Homocedasticidad

res1=summ$residuals
fit1=fitted(modp1)
dffp1=data.frame(res1,fit1)


ggplot(data = dffp, aes(x = fit1, y = res1)) +
  geom_point()
```




```
leveneTest(res1~BL*Nube*Solicitudes,data = base)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
```

```
##           Df F value                Pr(>F)
## group     11  115.79 < 0.00000000000000022 ***
##       13108
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Normalidad

ks.test(res1,"pnorm")
```

```
## Warning in ks.test.default(res1, "pnorm"): ties should not be present for the
## Kolmogorov-Smirnov test
```

```
##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  res1
## D = 0.27491, p-value < 0.00000000000000022
## alternative hypothesis: two-sided
```

```r
residuos1=as.data.frame(res1)
res.stnd1 = scale(residuos1$res1) # <- Residuos estandarizados.
ggplot(data = residuos1, mapping = aes(sample = res.stnd1 )) + stat_qq_point() +
      ylab("Residuos Estandarizados") + xlab("Norm Quantiles") + ggtitle("Normal QQ Plot")  + labs(capti
```

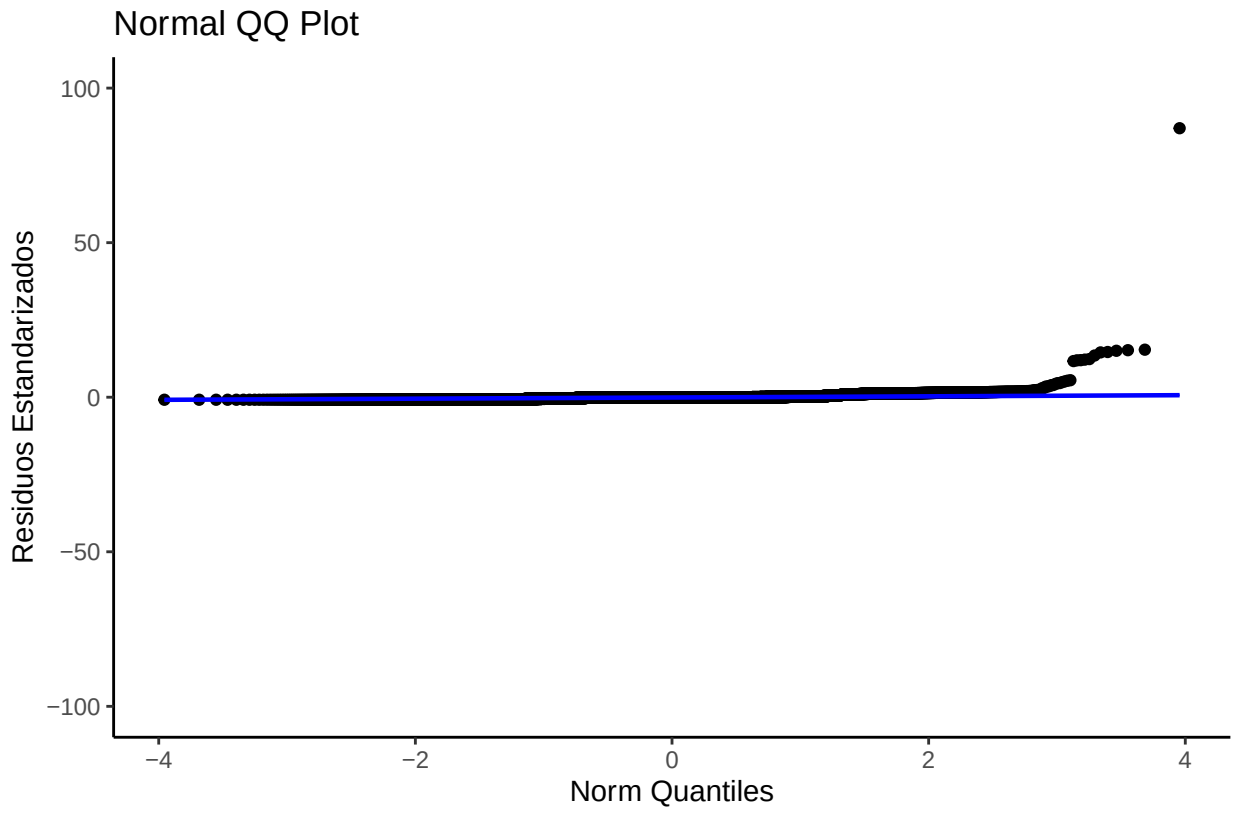


Grafico 1

```r
summary(modp1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##  Family: Gamma  ( log )
## Formula: TiempoT ~ BL * Nube * Solicitudes + (1 | Bloque)
##    Data: baseB
##
##      AIC      BIC   logLik deviance df.resid
##  45608.2  45713.0 -22790.1  45580.2    13106
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.2321 -0.3021 -0.0300  0.0572 24.1009
##
## Random effects:
##  Groups   Name        Variance  Std.Dev.
##  Bloque   (Intercept) 0.0005369 0.02317
##  Residual             0.5516859 0.74276
## Number of obs: 13120, groups:  Bloque, 20
##
## Fixed effects:
##                                Estimate Std. Error t value            Pr(>|z|)
## (Intercept)                    -2.44568    0.07366 -33.202 <0.0000000000000002
## BLML                            5.39638    0.10351  52.136 <0.0000000000000002
## NubeLocal                       1.20012    0.10339  11.608 <0.0000000000000002
## Solicitudes100                 -0.01603    0.07834  -0.205               0.8379
## Solicitudes500                  0.03780    0.07587   0.498               0.6183
## BLML:NubeLocal                  0.34706    0.14637   2.371               0.0177
## BLML:Solicitudes100            -0.18419    0.10929  -1.685               0.0919
## BLML:Solicitudes500            -0.24000    0.10515  -2.282               0.0225
## NubeLocal:Solicitudes100        0.01699    0.10914   0.156               0.8763
## NubeLocal:Solicitudes500       -0.03266    0.10503  -0.311               0.7558
## BLML:NubeLocal:Solicitudes100   0.19333    0.15450   1.251               0.2108
## BLML:NubeLocal:Solicitudes500   0.23572    0.14869   1.585               0.1129
##
## (Intercept)                    ***
## BLML                           ***
## NubeLocal                      ***
## Solicitudes100
## Solicitudes500
## BLML:NubeLocal                 *
## BLML:Solicitudes100            .
## BLML:Solicitudes500            *
## NubeLocal:Solicitudes100
## NubeLocal:Solicitudes500
## BLML:NubeLocal:Solicitudes100
## BLML:NubeLocal:Solicitudes500
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) BLML   NubLcl Slc100 Slc500 BLML:NbL BLML:S1 BLML:S5 NL:S10
```

```
## BLML        -0.701
## NubeLocal   -0.702  0.500
## Solictds100 -0.940  0.659  0.660
## Solictds500 -0.971  0.680  0.681  0.913
## BLML:NubLcl  0.496 -0.708 -0.707 -0.466 -0.481
## BLML:Slc100  0.663 -0.947 -0.474 -0.696 -0.644  0.671
## BLML:Slc500  0.690 -0.984 -0.492 -0.649 -0.691  0.697    0.933
## NbLcl:Sl100  0.665 -0.474 -0.947 -0.697 -0.645  0.670    0.500   0.467
## NbLcl:Sl500  0.691 -0.492 -0.984 -0.650 -0.692  0.696    0.466   0.500   0.933
## BLML:NL:S10 -0.469  0.671  0.670  0.492  0.456 -0.947   -0.708  -0.660  -0.707
## BLML:NL:S50 -0.488  0.697  0.696  0.459  0.489 -0.984   -0.660  -0.708  -0.659
##             NL:S50 BLML:NL:S1
## BLML
## NubeLocal
## Solictds100
## Solictds500
## BLML:NubLcl
## BLML:Slc100
## BLML:Slc500
## NbLcl:Sl100
## NbLcl:Sl500
## BLML:NL:S10 -0.659
## BLML:NL:S50 -0.707  0.933
## optimizer (Nelder_Mead) convergence code: 0 (OK)
## Model failed to converge with max|grad| = 0.0180776 (tol = 0.002, component 1)
```

Ningún modelo con los datos como estaban resultó satisfactorio y puede llevarnos a cometer errores.

Se decide explorar alternativas un enfoque diferente.

Se debe promediar la información de las URL de cada subparcela(lo que significa es que dentro de cada bloque de 10,100,500 URL se toma el promedio por tratamiento, por lo que cada bloque ahora tienen 4 respuestas, más abajo imprimo la base para que la vea a detalle). Se obtiene tanto el promedio como la mediana. Voy a modelar el promedio, pero hagan lo mismo con la mediana a ver.

```
rm(base1);rm(base2)
save(base,file = "base.Rdata")
base2 <- base %>%
  group_by(Bloque, BL,Nube, Solicitudes) %>%
  summarise(promedio=mean(TiempoT),mediana=median(TiempoT))
```

```
## `summarise()` has grouped output by 'Bloque', 'BL', 'Nube'. You can override
## using the `.groups` argument.
```

```
base2$Bloque=as.factor(base2$Bloque)
base2
```

```
## # A tibble: 80 x 6
## # Groups:   Bloque, BL, Nube [80]
##    Bloque BL    Nube  Solicitudes promedio  mediana
##    <fct>  <fct> <fct> <fct>          <dbl>    <dbl>
##  1 N1001  BL    Nube  100           0.0772   0.0688
##  2 N1001  BL    Local 100           0.300    0.110
##  3 N1001  ML    Nube  100          15.6     15.6
```

```
##  4 N1001  ML    Local 100          88.9     87.1
##  5 N1002  BL    Nube  100           0.0796   0.0717
##  6 N1002  BL    Local 100           0.260    0.0825
##  7 N1002  ML    Nube  100          15.8     15.7
##  8 N1002  ML    Local 100         102.     102.
##  9 N1003  BL    Nube  100           0.0760   0.0670
## 10 N1003  BL    Local 100           0.261    0.0602
## # i 70 more rows
```

```
basemod=base2
modp0=lmer(promedio~BL*Nube*Solicitudes+(1|Bloque),data = base2)
drop1(modp0,test = "Chisq")
```

```
## Single term deletions
##
## Model:
## promedio ~ BL * Nube * Solicitudes + (1 | Bloque)
##                      npar    AIC    LRT Pr(Chi)
## <none>                    435.77
## BL:Nube:Solicitudes    2 434.05 2.2844  0.3191
```

```
modp1=lmer(promedio~BL*Nube+Nube*Solicitudes+BL*Solicitudes+(1|Bloque),data = base2)
drop1(modp1,test = "Chisq")
```

```
## boundary (singular) fit: see help('isSingular')
```

```
## Single term deletions
##
## Model:
## promedio ~ BL * Nube + Nube * Solicitudes + BL * Solicitudes +
##     (1 | Bloque)
##                   npar    AIC     LRT             Pr(Chi)
## <none>                 434.05
## BL:Nube              1 715.83 283.773 <0.0000000000000002 ***
## Nube:Solicitudes     2 432.25   2.199              0.3330
## BL:Solicitudes       2 431.28   1.228              0.5413
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modp1=lmer(promedio~BL*Nube+Nube*Solicitudes+(1|Bloque),data = base2)
drop1(modp1,test = "Chisq")
```

```
## boundary (singular) fit: see help('isSingular')
```

```
## Warning in optwrap(optimizer, devfun, x@theta, lower = x@lower, calc.derivs =
## TRUE, : convergence code 3 from bobyqa: bobyqa -- a trust region step failed to
## reduce q
```

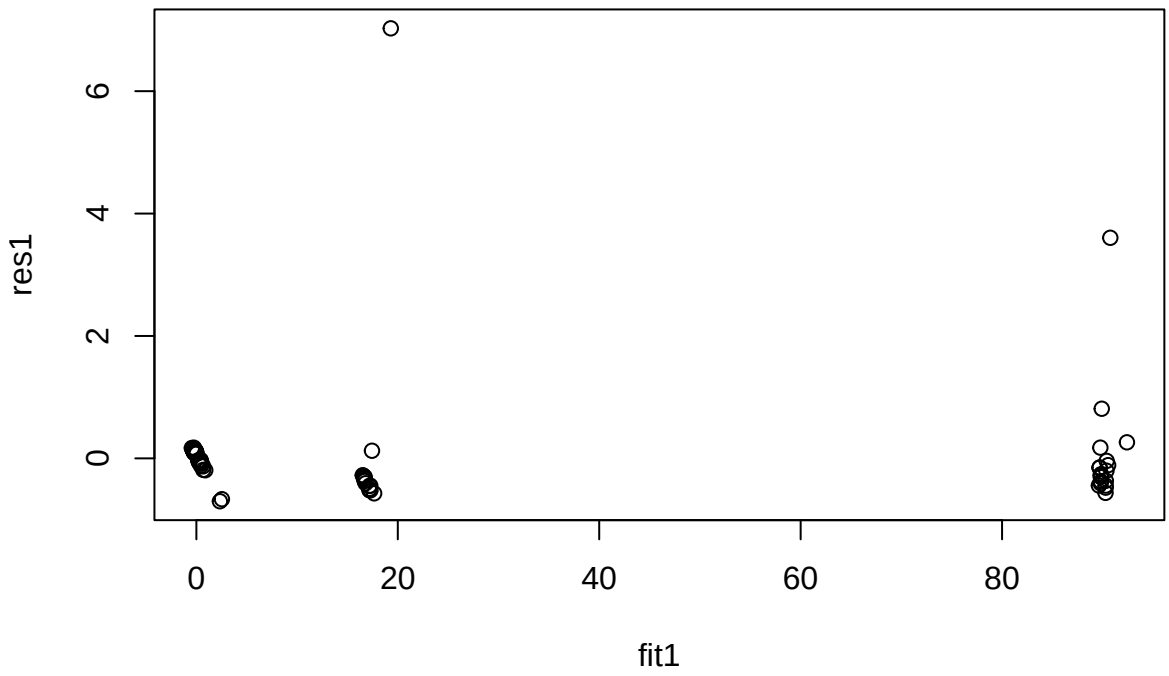
```
## Single term deletions
##
## Model:
```

```
## promedio ~ BL * Nube + Nube * Solicitudes + (1 | Bloque)
##                   npar    AIC     LRT             Pr(Chi)
## <none>                 431.28
## BL:Nube              1 711.86 282.578 <0.0000000000000002 ***
## Nube:Solicitudes     2 429.44   2.155              0.3404
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
modp1=lmer(promedio~BL*Nube+Solicitudes+(1|Bloque),data = base2)
drop1(modp1,test = "Chisq")
```

```
## boundary (singular) fit: see help('isSingular')

## Single term deletions
##
## Model:
## promedio ~ BL * Nube + Solicitudes + (1 | Bloque)
##             npar    AIC     LRT             Pr(Chi)
## <none>           429.44
## Solicitudes    2 426.36   0.923              0.6302
## BL:Nube        1 707.92 280.483 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summ1=summary(modp1)

res1=summ1$residuals
fit1=fitted(modp1)
dffp=data.frame(res1,fit1)
plot(fit1,res1)
```




```
leveneTest(promedio~BL*Nube*Solicitudes,data = base2)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group 11  0.8718 0.5713
##       68
```

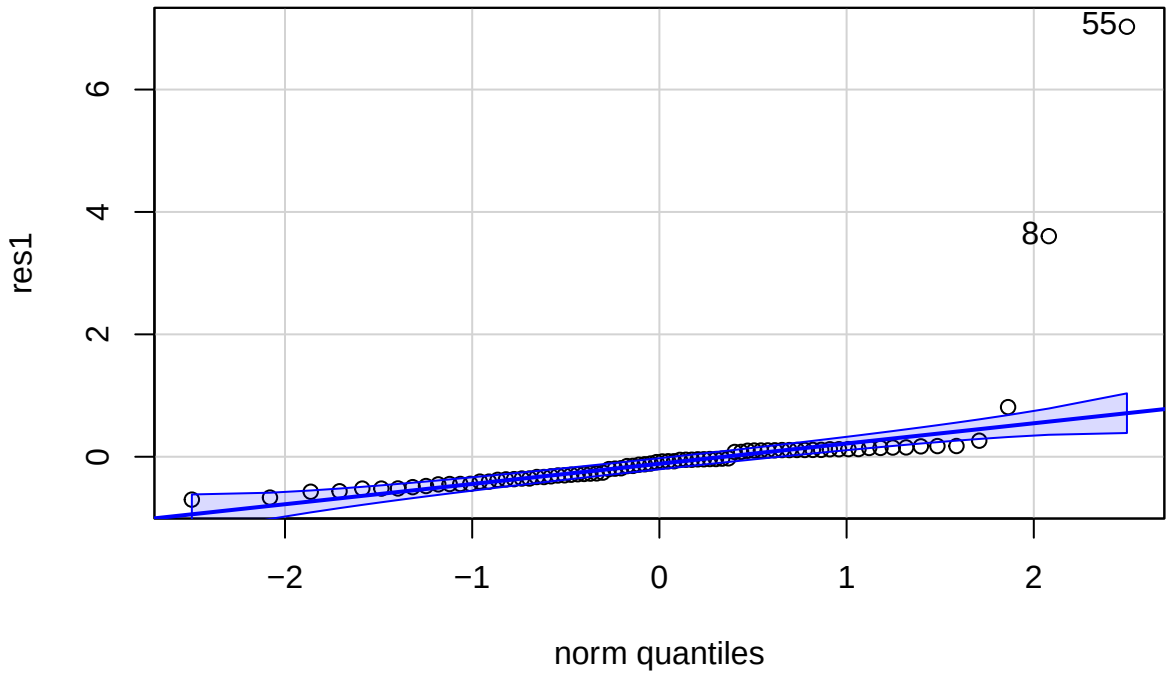
```
leveneTest(res1~BL*Nube*Solicitudes,data = base2)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##       Df F value Pr(>F)
## group 11  0.7738 0.6645
##       68
```

```
ks.test(res1,"pnorm")
```

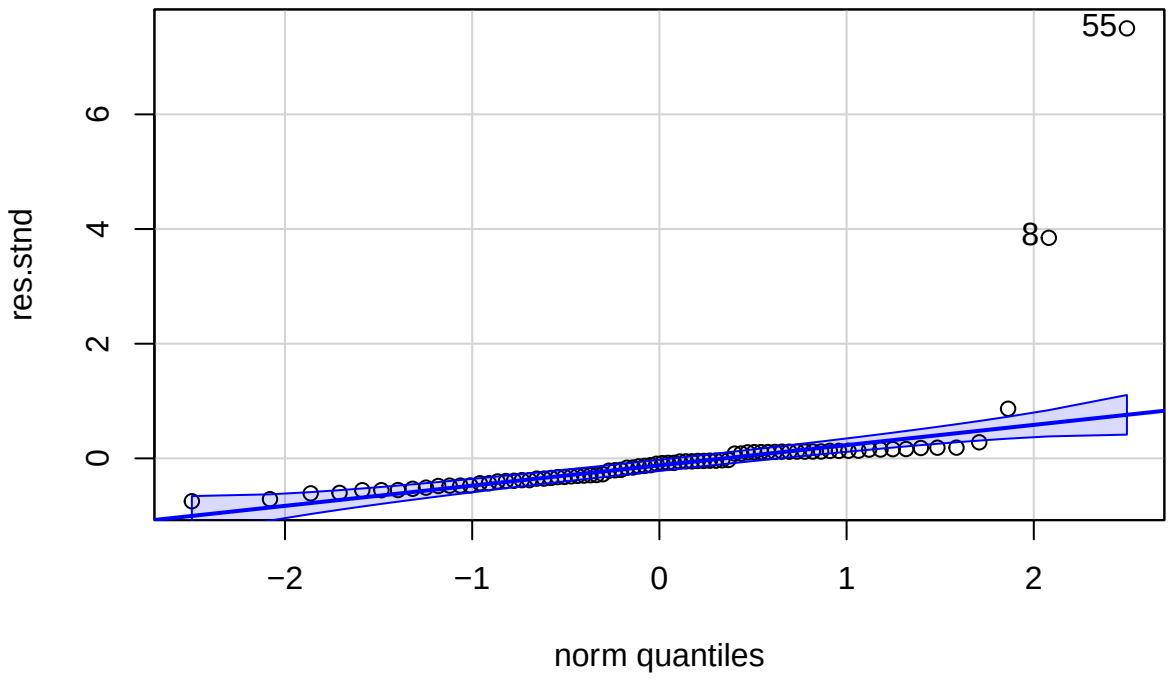
```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  res1
## D = 0.38043, p-value = 0.00000000006436
## alternative hypothesis: two-sided
```

```
residuos=as.data.frame(res1)
res.stnd = scale(residuos$res1) # <- Residuos estandarizados.
qqPlot(res1)
```




```
## [1] 55  8
```

```
qqPlot(res.stnd)
```




```
## [1] 55  8
```

Aquí pasan varias cosas, una es que hay dos parcelas con valores muy distintos al resto. Se puede quitar esas parcelas o asignarle la mediana. Sería al tratamiento ML Local de N1002, y N107 ML Nube. La otra es que aunque con Levene no se rechaza la hipótesis, se ve en el gráfico de ajustados y residuos que no hay homoscedasticidad. La opción es quitar esas dos parcelas y agregar pesos.

De nuevo, esto es pura estadística.

```
base2=base2[-which(base2$Bloque=="N107"|base2$Bloque=="N1002"),]
v=tapply(base2$promedio,list(base2$BL,base2$Nube,base2$Solicitudes),var);v
```

```
## , , 10
##
##            Nube       Local
## BL 0.0003839963 0.001186675
## ML 0.6974672107 0.507712146
##
## , , 100
##
##            Nube        Local
## BL 0.0001304958 0.0006774042
## ML 0.0053223055 0.0419083587
##
## , , 500
```

```
##
##             Nube        Local
## BL 0.00009561048 0.0001607724
## ML 0.00459759854 2.7006589315
```

```
pesos=data.frame(Solicitudes=rep(c(10,100,500),each=4),
                 Nube=rep(rep(c("Nube","Local"),each=2),3),
                 BL=rep(c("BL","ML"),6),w=as.vector(v))
base3=merge(base2,pesos,by=c("BL","Nube","Solicitudes"))
basemod=base3
modp1=lmer(promedio~BL*Nube+Solicitudes+(1|Bloque),weights = 1/w,data = basemod)
basemod
```

```
##    BL  Nube Solicitudes Bloque   promedio  mediana             w
## 1   BL Local          10   N101  0.2757800  0.20360 0.00118667478
## 2   BL Local          10   N102  0.3399800  0.10310 0.00118667478
## 3   BL Local          10   N103  0.2662000  0.08705 0.00118667478
## 4   BL Local          10   N104  0.2501300  0.04275 0.00118667478
## 5   BL Local          10   N105  0.2493300  0.07865 0.00118667478
## 6   BL Local          10   N106  0.2885400  0.08785 0.00118667478
## 7   BL Local          10   N108  0.2375500  0.04255 0.00118667478
## 8   BL Local         100  N1001  0.3000680  0.11050 0.00067740422
## 9   BL Local         100  N1003  0.2605620  0.06025 0.00067740422
## 10  BL Local         100  N1004  0.2795020  0.08455 0.00067740422
## 11  BL Local         100  N1005  0.2806420  0.05555 0.00067740422
## 12  BL Local         100  N1006  0.3362070  0.15575 0.00067740422
## 13  BL Local         100  N1008  0.3021260  0.07120 0.00067740422
## 14  BL Local         500  N5001  0.2956792  0.11275 0.00016077243
## 15  BL Local         500  N5002  0.2793432  0.08210 0.00016077243
## 16  BL Local         500  N5003  0.2790646  0.08635 0.00016077243
## 17  BL Local         500  N5005  0.3085038  0.07100 0.00016077243
## 18  BL Local         500  N5007  0.2838494  0.09855 0.00016077243
## 19  BL  Nube          10   N101  0.0834000  0.09405 0.00038399630
## 20  BL  Nube          10   N105  0.0720300  0.07170 0.00038399630
## 21  BL  Nube          10   N102  0.0879100  0.09185 0.00038399630
## 22  BL  Nube          10   N106  0.0899800  0.08740 0.00038399630
## 23  BL  Nube          10   N103  0.1245400  0.10000 0.00038399630
## 24  BL  Nube          10   N108  0.0843200  0.08240 0.00038399630
## 25  BL  Nube          10   N104  0.0616100  0.05000 0.00038399630
## 26  BL  Nube         100  N1001  0.0771940  0.06880 0.00013049576
## 27  BL  Nube         100  N1006  0.1027040  0.11560 0.00013049576
## 28  BL  Nube         100  N1003  0.0759850  0.06695 0.00013049576
## 29  BL  Nube         100  N1008  0.0956690  0.07845 0.00013049576
## 30  BL  Nube         100  N1004  0.0902330  0.08015 0.00013049576
## 31  BL  Nube         100  N1005  0.0767650  0.06895 0.00013049576
## 32  BL  Nube         500  N5001  0.0839990  0.07575 0.00009561048
## 33  BL  Nube         500  N5007  0.0851676  0.07570 0.00009561048
## 34  BL  Nube         500  N5002  0.1059024  0.07685 0.00009561048
## 35  BL  Nube         500  N5003  0.1017220  0.07885 0.00009561048
## 36  BL  Nube         500  N5005  0.0921758  0.08875 0.00009561048
## 37  ML Local          10   N104 89.7172000 86.99765 0.50771214628
## 38  ML Local          10   N101 89.1341600 87.41360 0.50771214628
## 39  ML Local          10   N105 90.1769200 86.72010 0.50771214628
## 40  ML Local          10   N102 88.7559400 86.85750 0.50771214628
```

```
## 41 ML Local          10   N106 88.4694000 86.23995 0.50771214628
## 42 ML Local          10   N103 88.8637400 88.93860 0.50771214628
## 43 ML Local          10   N108 90.2539500 88.25660 0.50771214628
## 44 ML Local         100  N1004 88.9890430 87.00100 0.04190835874
## 45 ML Local         100  N1005 88.4860580 87.12620 0.04190835874
## 46 ML Local         100  N1001 88.9464110 87.05910 0.04190835874
## 47 ML Local         100  N1006 88.5997430 87.06400 0.04190835874
## 48 ML Local         100  N1003 88.8317240 87.21475 0.04190835874
## 49 ML Local         100  N1008 88.9135190 87.53985 0.04190835874
## 50 ML Local         500  N5002 89.1937660 87.19460 2.70065893149
## 51 ML Local         500  N5003 89.2072432 87.60185 2.70065893149
## 52 ML Local         500  N5005 88.1979706 86.80130 2.70065893149
## 53 ML Local         500  N5001 90.3217212 87.51760 2.70065893149
## 54 ML Local         500  N5007 92.4981764 87.81155 2.70065893149
## 55 ML  Nube          10   N101 15.5423700 15.44880 0.69746721070
## 56 ML  Nube          10   N102 15.5997800 15.49045 0.69746721070
## 57 ML  Nube          10   N103 15.5319800 15.56240 0.69746721070
## 58 ML  Nube          10   N104 15.8376100 15.87890 0.69746721070
## 59 ML  Nube          10   N105 17.8350600 15.74105 0.69746721070
## 60 ML  Nube          10   N106 15.7315500 15.81485 0.69746721070
## 61 ML  Nube          10   N108 15.6216700 15.71720 0.69746721070
## 62 ML  Nube         100  N1001 15.5815270 15.56910 0.00532230548
## 63 ML  Nube         100  N1003 15.6532180 15.47775 0.00532230548
## 64 ML  Nube         100  N1004 15.5822280 15.51980 0.00532230548
## 65 ML  Nube         100  N1005 15.6296250 15.54350 0.00532230548
## 66 ML  Nube         100  N1006 15.7406810 15.59615 0.00532230548
## 67 ML  Nube         100  N1008 15.5308640 15.45940 0.00532230548
## 68 ML  Nube         500  N5001 15.6383646 15.54975 0.00459759854
## 69 ML  Nube         500  N5002 15.6447546 15.49655 0.00459759854
## 70 ML  Nube         500  N5003 15.6729752 15.56420 0.00459759854
## 71 ML  Nube         500  N5005 15.6357696 15.54985 0.00459759854
## 72 ML  Nube         500  N5007 15.5000062 15.39410 0.00459759854
```

```
drop1(modp1,test = "Chisq")
```

```
## boundary (singular) fit: see help('isSingular')

## Single term deletions
##
## Model:
## promedio ~ BL * Nube + Solicitudes + (1 | Bloque)
##             npar     AIC    LRT             Pr(Chi)
## <none>           -152.82
## Solicitudes    2 -154.82   2.00              0.3679
## BL:Nube        1  518.49 673.31 <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(modp1)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: promedio ~ BL * Nube + Solicitudes + (1 | Bloque)
```

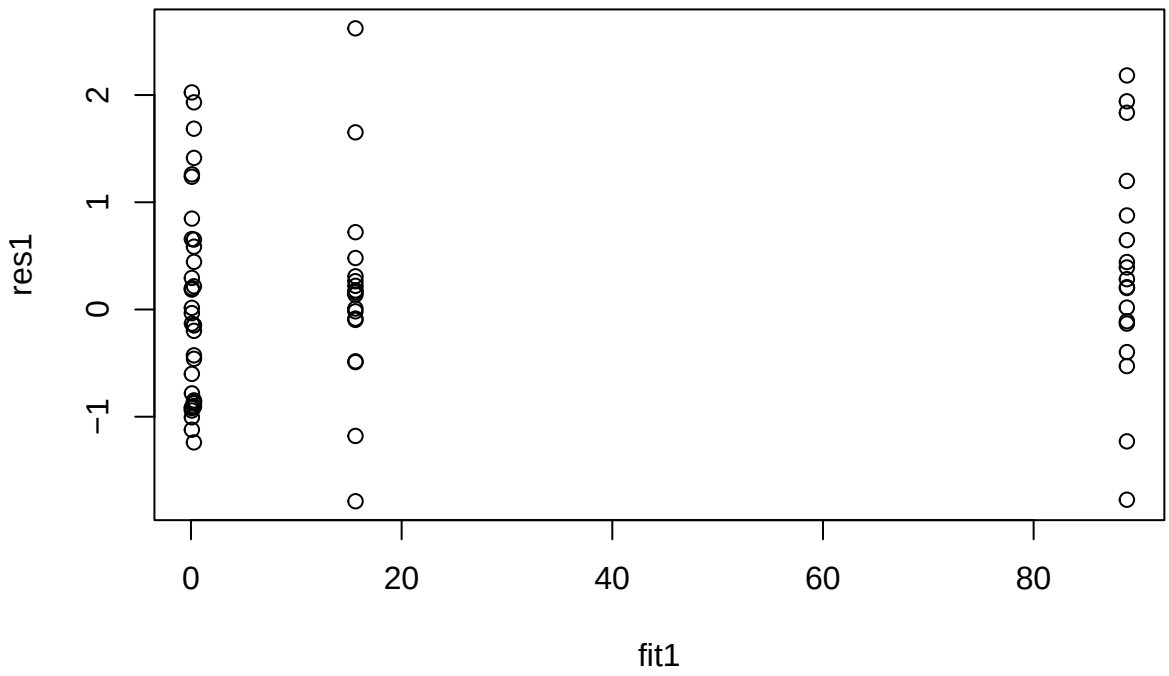
```
##    Data: basemod
## Weights: 1/w
##
## REML criterion at convergence: -124.3
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -1.78812 -0.49989  0.07805  0.60104  2.62177
##
## Random effects:
##  Groups   Name        Variance    Std.Dev.
##  Bloque   (Intercept) 0.000002334 0.001528
##  Residual             1.028944431 1.014369
## Number of obs: 72, groups:  Bloque, 18
##
## Fixed effects:
##                  Estimate Std. Error t value
## (Intercept)     0.084086   0.006700  12.549
## BLML           15.530070   0.021738 714.422
## NubeLocal       0.196843   0.005724  34.392
## Solicitudes100  0.003818   0.007880   0.485
## Solicitudes500  0.009162   0.007496   1.222
## BLML:NubeLocal 73.039826   0.083586 873.826
##
## Correlation of Fixed Effects:
##             (Intr) BLML   NubLcl Slc100 Slc500
## BLML        -0.016
## NubeLocal   -0.209  0.070
## Solictds100 -0.825 -0.050  0.059
## Solictds500 -0.835 -0.043 -0.097  0.723
## BLML:NubLcl  0.009 -0.259 -0.075 -0.010  0.022
```

```
confint(modp1)
```

```
## Computing profile confidence intervals ...

## Warning in nextpar(mat, cc, i, delta, lowcut, upcut): Last two rows have
## identical or NA .zeta values: using minstep

## Warning in FUN(X[[i]], ...): non-monotonic profile for .sig01

## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for (Intercept)

## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for BLML

## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for NubeLocal

## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for Solicitudes100
```

```
## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for Solicitudes500

## Warning in profile.merMod(object, which = parm, signames = oldNames, ...):
## non-monotonic profile for BLML:NubeLocal

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## .sig01: falling back to linear interpolation

## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## (Intercept): falling back to linear interpolation

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## BLML: falling back to linear interpolation

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## NubeLocal: falling back to linear interpolation

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## Solicitudes100: falling back to linear interpolation

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## Solicitudes500: falling back to linear interpolation

## Warning in confint.thpr(pp, level = level, zeta = zeta): bad spline fit for
## BLML:NubeLocal: falling back to linear interpolation

##                        2.5 %       97.5 %
## .sig01           0.000000000  0.010552234
## .sigma           0.833740900  1.156960502
## (Intercept)      0.084082845  0.084293550
## BLML            15.530057682 15.530769672
## NubeLocal        0.196840497  0.197022577
## Solicitudes100   0.003813710  0.004082711
## Solicitudes500   0.009158496  0.009394648
## BLML:NubeLocal 73.039781803 73.042460892
```

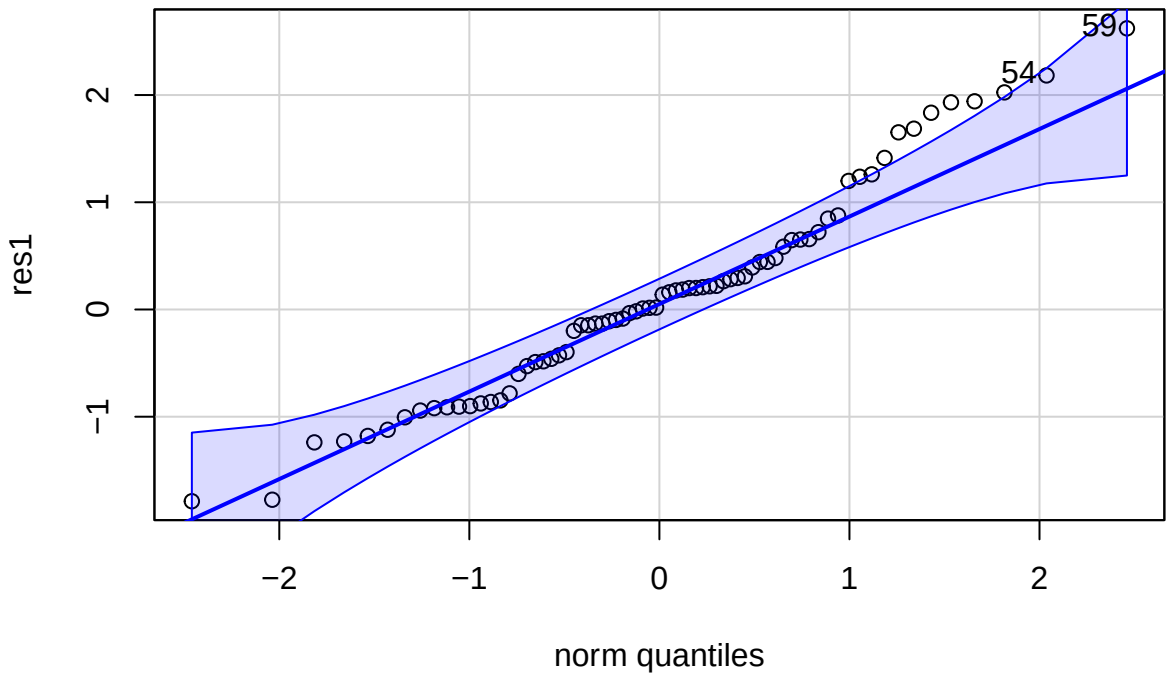
```
summ1=summary(modp1)

res1=summ1$residuals
fit1=fitted(modp1)
dffp=data.frame(res1,fit1)
plot(fit1,res1)
```




```
#Esto ya no se revisa porque se trabaja considerando heteroscedasticidad
# leveneTest(promedio~BL*Nube*Solicitudes,data = basemod)
# leveneTest(res1~BL*Nube*Solicitudes,data = basemod)

ks.test(res1,"pnorm")
```

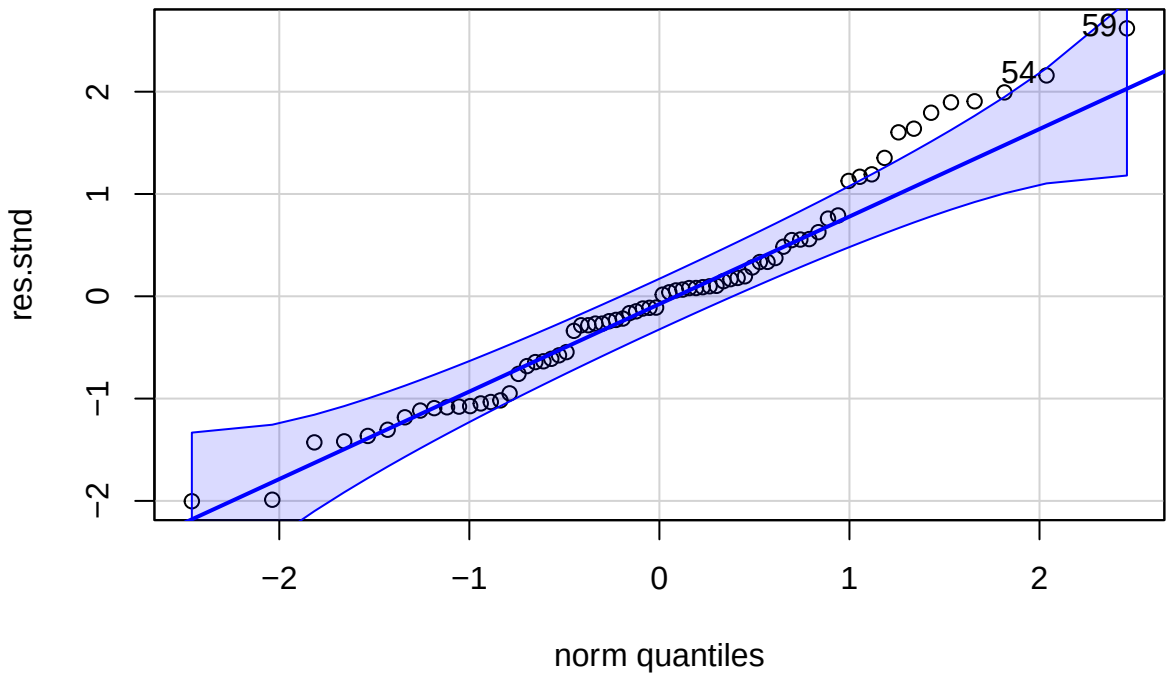
```
##
##  Exact one-sample Kolmogorov-Smirnov test
##
## data:  res1
## D = 0.10812, p-value = 0.3442
## alternative hypothesis: two-sided
```

```
residuos=as.data.frame(res1)
res.stnd = scale(residuos$res1) # <- Residuos estandarizados.
qqPlot(res1)
```




```
## [1] 59 54
```

```
qqPlot(res.stnd)
```




```
## [1] 59 54
```

```
contlmer=emmeans (modp1,  pairwise~ BL | Nube, adjust="bonferroni");contlmer
```

```
## $emmeans
## Nube = Nube:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.08841 0.003246 216847212  0.08205  0.09477
##  ML 15.61848 0.021645   1057792 15.57606 15.66091
##
## Nube = Local:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.28526 0.005125 175942416  0.27521  0.29530
##  ML 88.85515 0.081036      5567 88.69629 89.01401
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
## Nube = Nube:
##  contrast estimate      SE      df   t.ratio p.value
##  BL - ML    -15.53 0.02175 1114645  -714.111  <.0001
##
## Nube = Local:
```

```
##  contrast estimate      SE      df   t.ratio p.value
##  BL - ML    -88.57 0.08120    5615 -1090.707  <.0001
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
```

Este de arriba sería el modelo final, hay normalidad, no hay homoscedasticidad pero se está considerando la heteroscedasticidad al incluir pesos y se hace una mejor estimación del error estándar para las comparaciones. Hay que calcular cotas inferiores y concluir con respecto a una diferencia relevante, porque ambas son significativas.

Este sería todo el procedimiento que se usó para llegar al modelo. El cual es el siguiente.

El modelo final

$$\mu_{ijk} = \mu + \alpha_i + \beta_j + \delta_k + (\alpha\beta)_{ij} + \nu_l$$

Donde:

- μ es la media general, es decir, sin tomar en cuenta ningún tratamiento.
- α_i es el efecto del nivel i del Factor BlackList (Método de detección)
- β_j es el efecto del nivel j del Factor Nube (Ubicación)
- δ_k es el efecto del nivel k del Factor Solicitudes
- $(\alpha\beta)_{ij}$ es el efecto de la interacción entre BlackList y Nube
- ν_l Es el efecto de la parcela. Es decir, el efecto que cada bloque de 10,100,500 trae sobre el modelo.

Ahora se hacen las pruebas de hipótesis

```
contlmer=emmeans (modp1,  pairwise~ BL | Nube, adjust="bonferroni");contlmer
```

```
## $emmeans
## Nube = Nube:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.08841 0.003246 216847212  0.08205  0.09477
##  ML 15.61848 0.021645   1057792 15.57606 15.66091
##
## Nube = Local:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.28526 0.005125 175942416  0.27521  0.29530
##  ML 88.85515 0.081036      5567 88.69629 89.01401
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
## Nube = Nube:
##  contrast estimate      SE      df   t.ratio p.value
##  BL - ML    -15.53 0.02175 1114645  -714.111  <.0001
##
## Nube = Local:
##  contrast estimate      SE      df   t.ratio p.value
```

```
##  BL - ML    -88.57 0.08120    5615 -1090.707  <.0001
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
```

Y se calculan los intervalos.

```
confint(contlmer)
```

```
## $emmeans
## Nube = Nube:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.08841 0.003246 216847212  0.08205  0.09477
##  ML 15.61848 0.021645   1057792 15.57606 15.66091
##
## Nube = Local:
##  BL   emmean       SE        df lower.CL upper.CL
##  BL  0.28526 0.005125 175942416  0.27521  0.29530
##  ML 88.85515 0.081036      5567 88.69629 89.01401
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
##
## $contrasts
## Nube = Nube:
##  contrast estimate      SE      df lower.CL upper.CL
##  BL - ML    -15.53 0.02175 1114645   -15.57   -15.49
##
## Nube = Local:
##  contrast estimate      SE      df lower.CL upper.CL
##  BL - ML    -88.57 0.08120    5615   -88.73   -88.41
##
## Results are averaged over the levels of: Solicitudes
## Degrees-of-freedom method: kenward-roger
## Confidence level used: 0.95
```

Más sobre los resultados en el Word.