Epidemiological exercises in R

Lucky Mehra

2019-08-02

Contents

Prerequisites

To run these exercises, you will need to install the latest version of R (https://cloud.r-project.org/) and RStudio (https://www.rstudio.com/products/rstudio/download/) on your computer. Please click on the above mentioned links to go to the download pages of R and RStudio.

These exercises are a work in progress, and are an attempt to translate SAS code written by Tim Todd into R.

Chapter 1

Exercise 4

1.1 Load packages

Here is the R code to download the required packages for this exercise.

```r
# install package manager 'pacman'
if (!require(pacman)){
  install.packages('pacman')
}
```

```
## Loading required package: pacman
```

Load the packages needed for this exercise:

```r
pacman::p_load(tidyverse,
       nlme,
       emmeans)
```

1.2 Import data

Our data is located in `ex4.csv` file, which can be found on my github repo. Import the data and create new variables using the code below.

```r
# import data
a <- read_csv("https://raw.githubusercontent.com/luckymehra/epidem-exercises/master/data/ex4.csv'
              col_types = cols(
                blk = col_factor(), # parse blk as a factor
```

Table 1.1: The first 6 rows of dataset *a*.

plot	t	blk	trt	pctsev	y	ystar	wt
101	0	1	2	9	0.09	-2.313635	0.0819
102	0	1	1	6	0.06	-2.751535	0.0564
103	0	1	3	2	0.02	-3.891820	0.0196
201	0	2	2	7	0.07	-2.586689	0.0651
202	0	2	3	5	0.05	-2.944439	0.0475
203	0	2	1	3	0.03	-3.476099	0.0291

```
                trt = col_factor() # parse trt as a factor
                ))

# create new variables
a$y <- a$pctsev/100
a$ystar <- log(a$y/(1-a$y))
a$wt <- a$y*(1-a$y)

# print the data
knitr::kable(head(a),
             caption = "The first 6 rows of dataset *a*.")
```

```
# get a glimpse of data
glimpse(a)
```

```
## Observations: 72
## Variables: 8
## $ plot   <dbl> 101, 102, 103, 201, 202, 203, 301, 302, 303, 401, 402, ...
## $ t      <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 7, 7, 7, 7, 7, 7, 7...
## $ blk    <fct> 1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 1, 1, 1, 2, 2, 2, 3...
## $ trt    <fct> 2, 1, 3, 2, 3, 1, 3, 2, 1, 1, 2, 3, 2, 1, 3, 2, 3, 1, 3...
## $ pctsev <dbl> 9, 6, 2, 7, 5, 3, 4, 2, 6, 1, 1, 4, 4, 6, 10, 2, 5, 3, ...
## $ y      <dbl> 0.09, 0.06, 0.02, 0.07, 0.05, 0.03, 0.04, 0.02, 0.06, 0...
## $ ystar  <dbl> -2.313635, -2.751535, -3.891820, -2.586689, -2.944439, ...
## $ wt     <dbl> 0.0819, 0.0564, 0.0196, 0.0651, 0.0475, 0.0291, 0.0384,...
```

1.3 First mixed model

1.3.1 Fit the model

Run the mixed model analysis using **nlme** package in R. The function used to fit the mixed model is called `lme()`.

```
# fit the model

mm_1 <- lme(ystar ~ trt*t, # fixed effects
            data = a,
            random = list(blk = ~ 1, plot = ~ 1), # random effects
            correlation = corAR1(form = (plot = ~ 1)), # specify that observations within a plot
            contrasts = list(trt = "contr.SAS"), # specify this option to get parameter estimates
            weights = ~ I(1/wt))

# output the summary
summary(mm_1)
```

```
## Linear mixed-effects model fit by REML
##  Data: a
##        AIC      BIC    logLik
##   210.5257 232.4222 -95.26285
##
## Random effects:
##  Formula: ~1 | blk
##         (Intercept)
## StdDev:   0.1887117
##
##  Formula: ~1 | plot %in% blk
##          (Intercept)  Residual
## StdDev: 4.604287e-05 0.2519511
##
## Correlation Structure: AR(1)
##  Formula: ~1 | blk/plot
##  Parameter estimate(s):
##        Phi
## 0.06205463
## Variance function:
##  Structure: fixed weights
##  Formula: ~I(1/wt)
## Fixed effects: ystar ~ trt * t
##                  Value Std.Error DF   t-value p-value
## (Intercept) -2.5689859 0.3629604 57 -7.077868  0.0000
## trt2        -0.1948084 0.5193013  6 -0.375136  0.7205
## trt1        -0.5406041 0.5136249  6 -1.052527  0.3331
## t            0.0992675 0.0142177 57  6.981964  0.0000
## trt2:t      -0.0221696 0.0202998 57 -1.092109  0.2794
## trt1:t       0.0437431 0.0212717 57  2.056398  0.0443
##  Correlation:
##        (Intr) trt2   trt1   t      trt2:t
## trt2   -0.652
## trt1   -0.658  0.459
## t      -0.888  0.621  0.627
## trt2:t  0.623 -0.924 -0.439 -0.700
## trt1:t  0.592 -0.413 -0.913 -0.667  0.466
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.1518915 -0.6900213 -0.4024653  0.4132408  2.7733450
##
## Number of Observations: 72
## Number of Groups:
##          blk plot %in% blk
##            4            12
```

```
# extract covariance parameter estimates
VarCorr(mm_1)
```

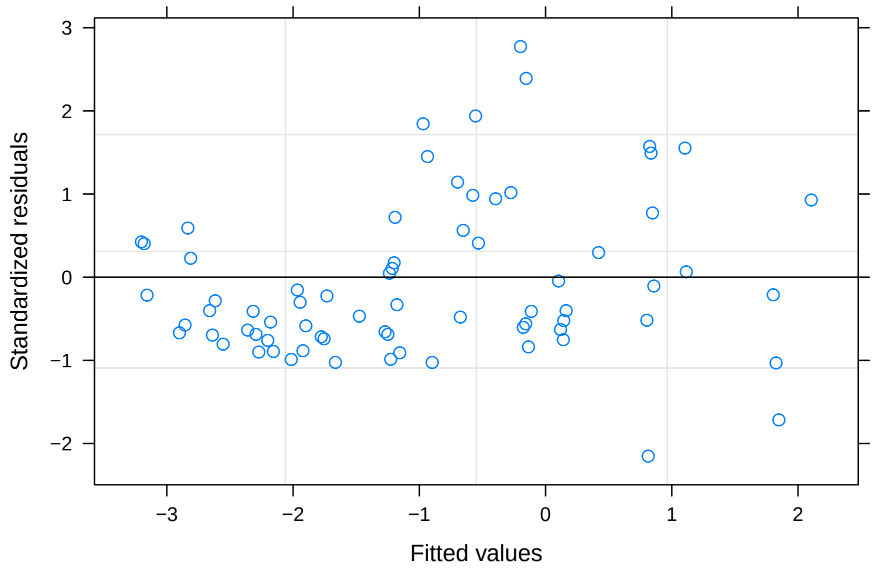
```
##             Variance     StdDev
## blk =       pdLogChol(1)
## (Intercept) 3.561212e-02 1.887117e-01
## plot =      pdLogChol(1)
## (Intercept) 2.119946e-09 4.604287e-05
## Residual    6.347936e-02 2.519511e-01
```

```
# extract type3 fixed effects anova
anova.lme(mm_1, type = 'marginal')
```

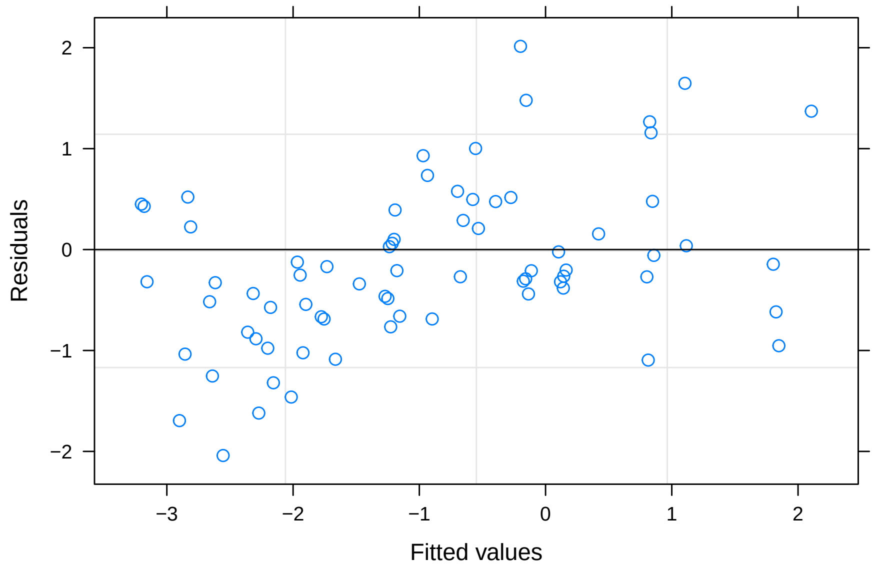
```
##             numDF denDF  F-value p-value
## (Intercept)     1    57 50.09622  <.0001
## trt             2     6  0.56135  0.5977
## t               1    57 48.74782  <.0001
## trt:t           2    57  4.80110  0.0118
```

1.3.2 Diagnostic plots

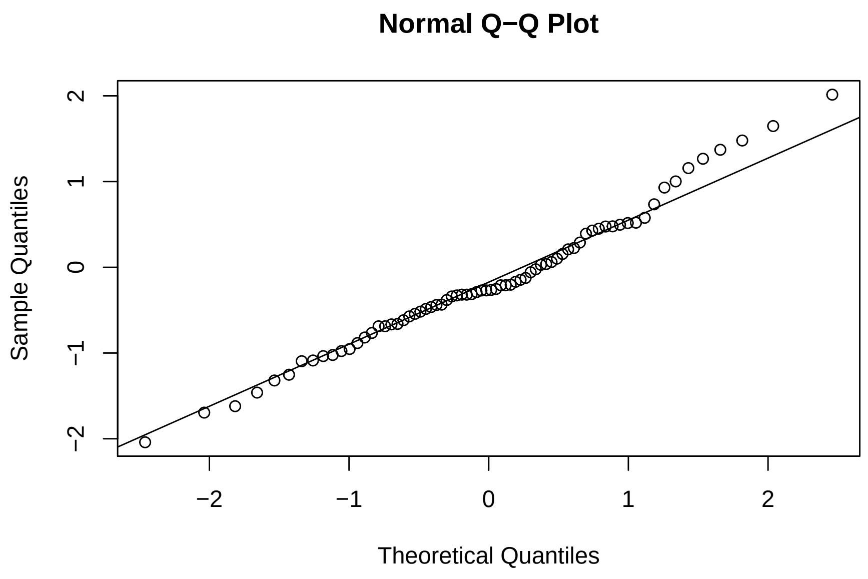
```
# pearson residuals vs. fitted values
plot(mm_1, resid(., type="pearson") ~ fitted(.), abline = 0)
```



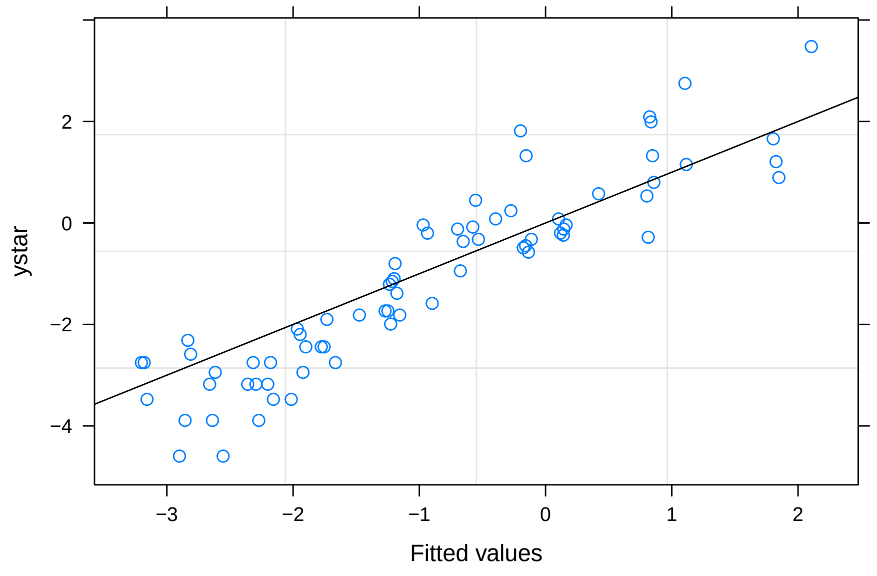
```
# standardaized residuals vs. fitted values
plot(mm_1, resid(., scaled=TRUE) ~ fitted(.), abline = 0)
```



```r
# qq plot
qqnorm(residuals(mm_1))
qqline(residuals(mm_1))
```



```r
#observed vs. fitted values
plot(mm_1, ystar ~ fitted(.), abline = c(0,1))
```



1.4 Second mixed model

1.4.1 Fit the model

Run the mixed model analysis using **nlme** package in R. The function used to fit the mixed model is called `lme()`. Here we will specify no intercept. We will also use **emmeans** package to get least squared means and contrasts.

```
# fit the model
#library(nlme)
mm_2 <- update(mm_1, fixed = ystar ~ - 1 + trt + trt:t) # update fixed effects in mm_1, -1 indica

# output the summary
summary(mm_2)
```

```
## Linear mixed-effects model fit by REML
##  Data: a
##        AIC      BIC    logLik
##   210.5257 232.4222 -95.26285
##
## Random effects:
##  Formula: ~1 | blk
##         (Intercept)
## StdDev:   0.1887117
##
##  Formula: ~1 | plot %in% blk
##          (Intercept)  Residual
## StdDev: 4.603147e-05 0.2519511
##
## Correlation Structure: AR(1)
##  Formula: ~1 | blk/plot
##  Parameter estimate(s):
##        Phi
## 0.06205463
## Variance function:
##  Structure: fixed weights
##  Formula: ~I(1/wt)
## Fixed effects: ystar ~ trt + trt:t - 1
##             Value Std.Error DF   t-value p-value
## trt2   -2.7637943 0.3944803  6 -7.006165   4e-04
## trt1   -3.1095900 0.3877657  6 -8.019250   2e-04
## trt3   -2.5689859 0.3629604  6 -7.077868   4e-04
## trt2:t  0.0770979 0.0144893 58  5.321034   0e+00
## trt1:t  0.1430106 0.0158560 58  9.019328   0e+00
## trt3:t  0.0992675 0.0142177 58  6.981964   0e+00
##  Correlation:
##        trt2   trt1   trt3   trt2:t trt1:t
## trt1    0.057
## trt3    0.062  0.065
## trt2:t -0.901  0.001  0.001
## trt1:t  0.001 -0.881 -0.002 -0.001
## trt3:t  0.000 -0.002 -0.888  0.000  0.002
##
## Standardized Within-Group Residuals:
##        Min         Q1        Med         Q3        Max
## -2.1518915 -0.6900213 -0.4024653  0.4132408  2.7733450
##
## Number of Observations: 72
## Number of Groups:
##           blk plot %in% blk
##             4            12
```

```
# extract covariance parameter estimates
VarCorr(mm_2)
```

```
##             Variance     StdDev
## blk =       pdLogChol(1)
## (Intercept) 3.561212e-02 1.887117e-01
## plot =      pdLogChol(1)
## (Intercept) 2.118896e-09 4.603147e-05
## Residual    6.347936e-02 2.519511e-01
```

```
# extract type3 fixed effects anova
anova.lme(mm_2, type = 'marginal')
```

```
##       numDF denDF  F-value p-value
## trt       3     6 48.57698   1e-04
## trt:t     3    58 52.74601  <.0001
```

```
# compare the slopes for different treatments
#library(emmeans)

emtrends(mm_2, pairwise ~ trt, var="t", adjust = "none")
```

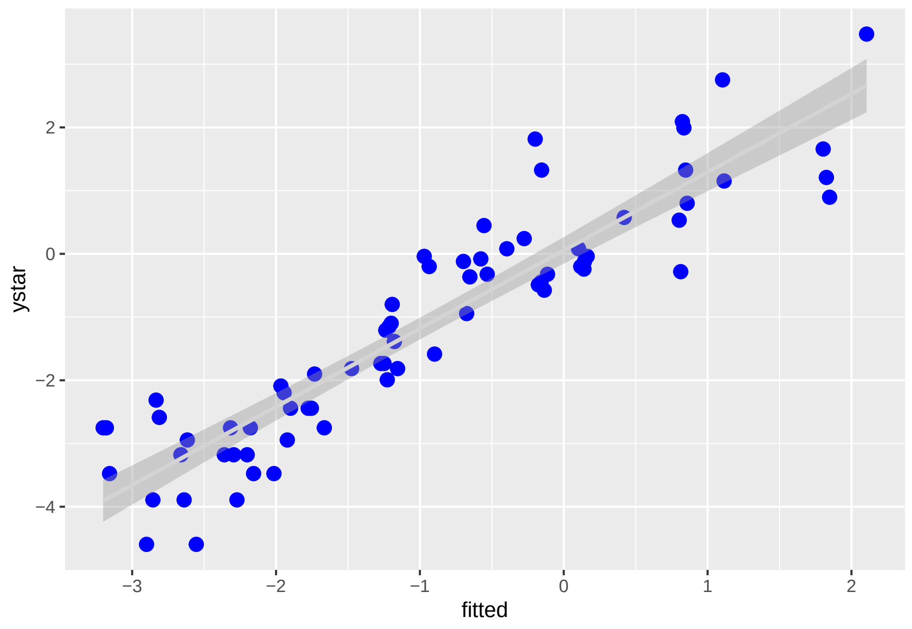
```
## $emtrends
##  trt t.trend     SE df lower.CL upper.CL
##  2    0.0771 0.0145 58   0.0481    0.106
##  1    0.1430 0.0159 58   0.1113    0.175
##  3    0.0993 0.0142 58   0.0708    0.128
##
## d.f. method: containment
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate     SE df t.ratio p.value
##  2 - 1     -0.0659 0.0215 58 -3.067  0.0033
##  2 - 3     -0.0222 0.0203 58 -1.092  0.2793
##  1 - 3      0.0437 0.0213 58  2.056  0.0443
```

```
# get the treatment difference at various time points
emmeans(mm_2, pairwise ~ trt|t, nesting = NULL, at = list(t = c(0, 7, 14, 21, 28, 35)), adjust =
```

```
## $emmeans
## t =  0:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -2.7638 0.394  6   -3.729  -1.7985
##  1   -3.1096 0.388  6   -4.058  -2.1608
##  3   -2.5690 0.363  6   -3.457  -1.6809
##
## t =  7:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -2.2241 0.306  6   -2.974  -1.4746
##  1   -2.1085 0.295  6   -2.830  -1.3873
##  3   -1.8741 0.278  6   -2.555  -1.1931
##
## t = 14:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -1.6844 0.229  6   -2.246  -1.1232
##  1   -1.1074 0.219  6   -1.644  -0.5712
##  3   -1.1792 0.207  6   -1.687  -0.6719
##
## t = 21:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -1.1447 0.179  6   -1.582  -0.7072
##  1   -0.1064 0.184  6   -0.556   0.3437
##  3   -0.4844 0.168  6   -0.896  -0.0726
##
## t = 28:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -0.6051 0.179  6   -1.042  -0.1680
##  1    0.8947 0.210  6    0.380   1.4095
##  3    0.2105 0.183  6   -0.237   0.6581
##
## t = 35:
##  trt  emmean    SE df lower.CL upper.CL
##  2   -0.0654 0.229  6   -0.626   0.4948
##  1    1.8958 0.282  6    1.207   2.5850
##  3    0.9054 0.242  6    0.314   1.4968
##
## d.f. method: containment
## Confidence level used: 0.95
##
## $contrasts
## t =  0:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1      0.3458 0.537  6  0.644  0.5435
##  2 - 3     -0.1948 0.519  6 -0.375  0.7205
##  1 - 3     -0.5406 0.514  6 -1.053  0.3331
##
## t =  7:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1     -0.1156 0.404  6 -0.286  0.7843
##  2 - 3     -0.3500 0.392  6 -0.893  0.4062
##  1 - 3     -0.2344 0.383  6 -0.613  0.5625
##
## t = 14:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1     -0.5770 0.288  6 -2.004  0.0919
##  2 - 3     -0.5052 0.279  6 -1.811  0.1201
##  1 - 3      0.0718 0.271  6  0.265  0.7996
##
## t = 21:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1     -1.0384 0.219  6 -4.739  0.0032
##  2 - 3     -0.6604 0.206  6 -3.204  0.0185
##  1 - 3      0.3780 0.211  6  1.794  0.1229
##
## t = 28:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1     -1.4998 0.242  6 -6.204  0.0008
##  2 - 3     -0.8156 0.218  6 -3.741  0.0096
##  1 - 3      0.6842 0.245  6  2.795  0.0314
##
## t = 35:
##  contrast estimate    SE df t.ratio p.value
##  2 - 1     -1.9611 0.338  6 -5.806  0.0011
##  2 - 3     -0.9707 0.305  6 -3.184  0.0190
##  1 - 3      0.9904 0.346  6  2.861  0.0288
```

1.4.2 Plot observed versus predicted model values

```
# add fitted and residuals in to a new dataset called b
b = cbind(a, resid = resid(mm_2), fitted = fitted(mm_2))

# fit linear regression
b.lm <- lm(ystar ~ fitted, data=b)

# plot using ggplot2 package
ggplot(b, aes(x=fitted, y = ystar)) +
  geom_point(color="blue", size = 3) +
  geom_smooth(method = lm, color = "lightgrey")
```


ystar
2
0
−2
−4
−3
−2
−1
0
1
2
fitted

Chapter 2

Exercise 9.4

2.1 Load packages

Here is the R code to download the required packages for this exercise.

```r
# install package manager 'pacman'
if (!require(pacman)){
  install.packages('pacman')
}
```

```
## Loading required package: pacman
```

```r
# load packages needed for this exercise
library(pacman)
p_load(tidyverse,
       lctools, # to calculate Moran's I
       spdep, # to calculate geary's c
       geoR, # to compute variogram
       gridExtra, # to stack plots
       gstat, automap, # packages for variogram model selection
       sp # need a function called `coordinates`
       )
```

2.2 Data

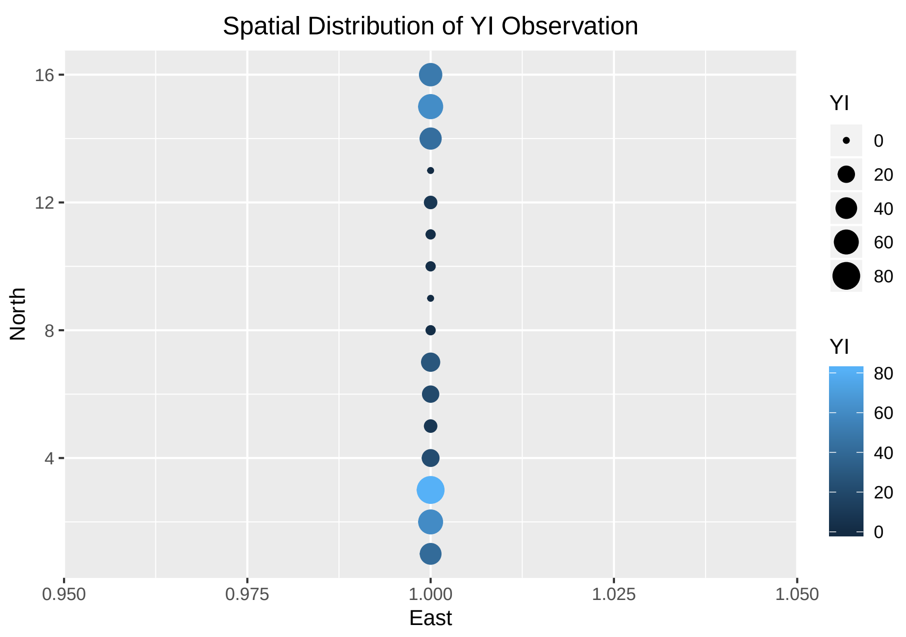
This is equivalent to data step in SAS. Here, the data is entered inside a function called `tibble`.

```r
# Enter data
a <- tibble(I = 1:16, YI = c(41, 60, 81, 22, 8, 20, 28, 2,
                              0, 2, 2, 8, 0, 43, 61, 50)) %>%
  # creat new variable East and North
  mutate(East = 1,
         North = I)

# print the data
a
```

```
## # A tibble: 16 x 4
##        I    YI  East North
##    <int> <dbl> <dbl> <int>
##  1     1    41     1     1
##  2     2    60     1     2
##  3     3    81     1     3
##  4     4    22     1     4
##  5     5     8     1     5
##  6     6    20     1     6
##  7     7    28     1     7
##  8     8     2     1     8
##  9     9     0     1     9
## 10    10     2     1    10
## 11    11     2     1    11
## 12    12     8     1    12
## 13    13     0     1    13
## 14    14    43     1    14
## 15    15    61     1    15
## 16    16    50     1    16
```

2.3 Autocorrelation statistics

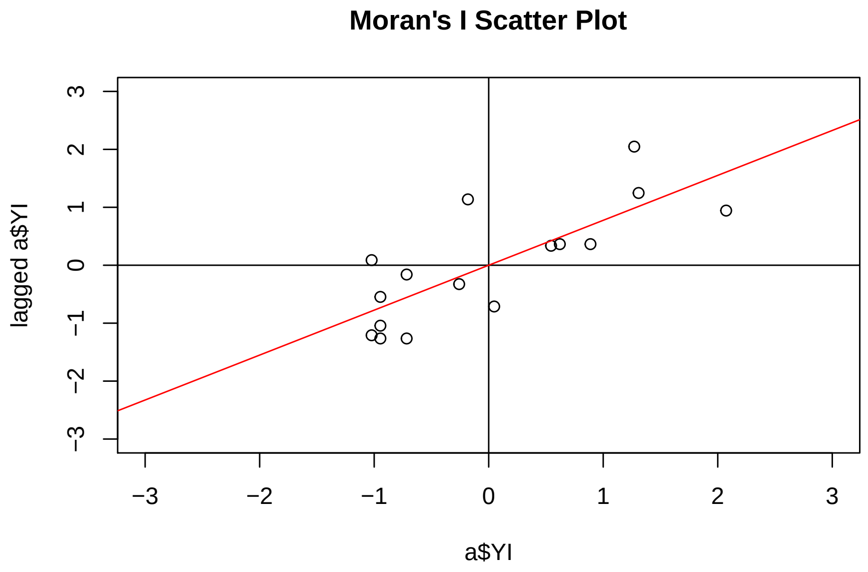
```r
# visualize the data
ggplot(data = a) +
  geom_point(mapping = aes(x = East, y = North, size = YI, color = YI)) +
  ggtitle("Spatial Distribution of YI Observation") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# calculate Moran's I
Coords <- a %>%
  dplyr::select(East, North)

mI <- moransI(Coords, Bandwidth = 1, a$YI)

# print Moran's I table
moran.table <- tribble(
  ~`Moran's I`, ~`Expected I`, ~`Z randomization`, ~`P value randomization`,
  #------------/--------------/-------------------/------------------------
  mI$Morans.I, mI$Expected.I,  mI$z.randomization, mI$p.value.randomization
  )

moran.table
```

```
## # A tibble: 1 x 4
##   `Moran's I` `Expected I` `Z randomization` `P value randomization`
##         <dbl>        <dbl>             <dbl>                   <dbl>
## 1       0.625      -0.0667              2.81                 0.00499
```

```
# create Moran's I scatter plot
l.moran <- l.moransI(Coords, Bandwidth = 1, a$YI)
```



```r
# calculate geary's c
Coords_num <- coordinates(Coords)

# create an object of class 'nb' so that it can be used with function from packege `sp
Coords_nb <- knn2nb(knearneigh(Coords_num))

# create a 'listw' object for use in the function `geary.test`
coords_listw <- nb2listw(Coords_nb)

gearyC <- geary.test(a$YI, coords_listw, alternative = "two.sided")
gearyC
```

```
##
##  Geary C test under randomisation
##
## data:  a$YI
## weights: coords_listw
##
## Geary C statistic standard deviate = 2.5826, p-value = 0.009806
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##        0.37085605        1.00000000        0.05934473
```

2.4 First variogram

We will use the package `geoR` to construct empricial variogram, and then draw them using package `ggplot2`.

```
v1 <- variog(coords = Coords_num, data = a$YI, breaks = seq(0.5, 15.5),
             max.dist = 11)
```

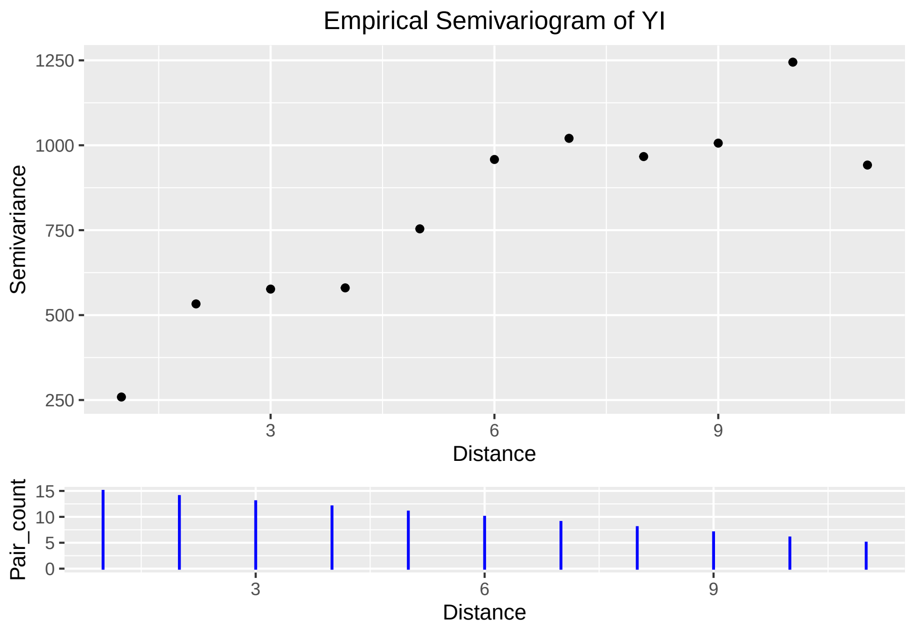
```
## variog: computing omnidirectional variogram
```

```
# extract data from object v1 for plotting
v1_plot_data <- cbind(v1$u, v1$v, v1$n) %>%
  as.data.frame() %>%
  dplyr::rename(Distance = V1,
                Semivariance = V2,
                Pair_count = V3)

# in the table below, gamma is semivariance
v1_plot_data
```

```
##    Distance Semivariance Pair_count
## 1         1     258.8333         15
## 2         2     533.0000         14
## 3         3     576.6154         13
## 4         4     580.1667         12
## 5         5     754.0000         11
## 6         6     958.2000         10
## 7         7    1020.4444          9
## 8         8     966.7500          8
## 9         9    1006.2857          7
## 10       10    1244.6667          6
## 11       11     941.8000          5
```

```
# plot variogram
v1_plot_vario <- ggplot(data = v1_plot_data) +
  geom_point(mapping = aes(x = Distance, y = Semivariance)) +
  ggtitle("Empirical Semivariogram of YI") +
  theme(plot.title = element_text(hjust = 0.5))

# plot pair counts
v1_plot_pair_count <- ggplot(data = v1_plot_data) +
  geom_col(mapping = aes(x = Distance, y = Pair_count), width = 0.01, color = "blue")

# stack two plots
```

```
grid.arrange(v1_plot_vario, v1_plot_pair_count,
             ncol = 1, heights = c(3, 1))
```



2.5 Second variogram

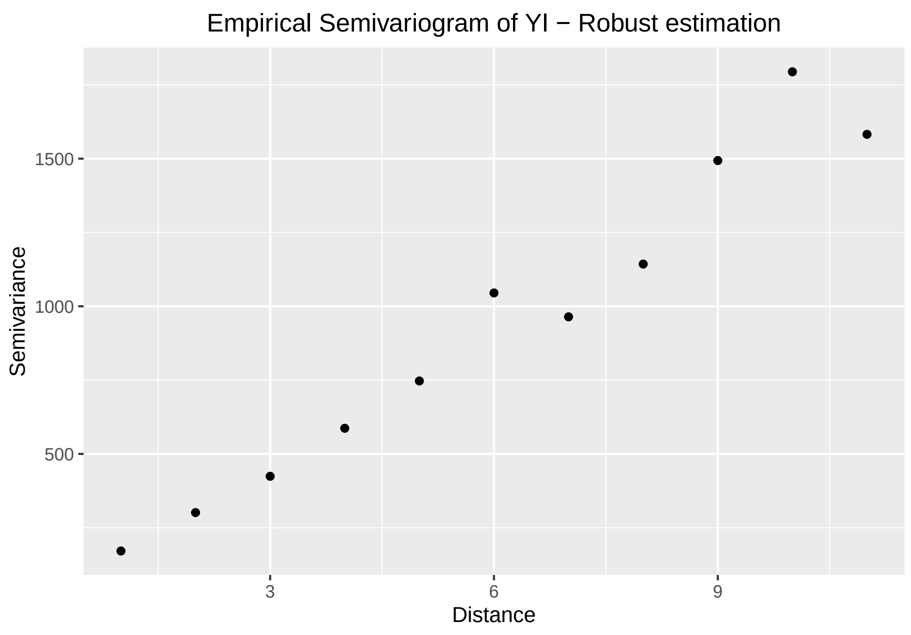
Plot robust and classical variogram together.

```
# fit robust variogram
v1_robust <- variog(coords = Coords_num, data = a$YI, breaks = seq(0.5, 15.5),
             max.dist = 11, estimator.type = "modulus")
```

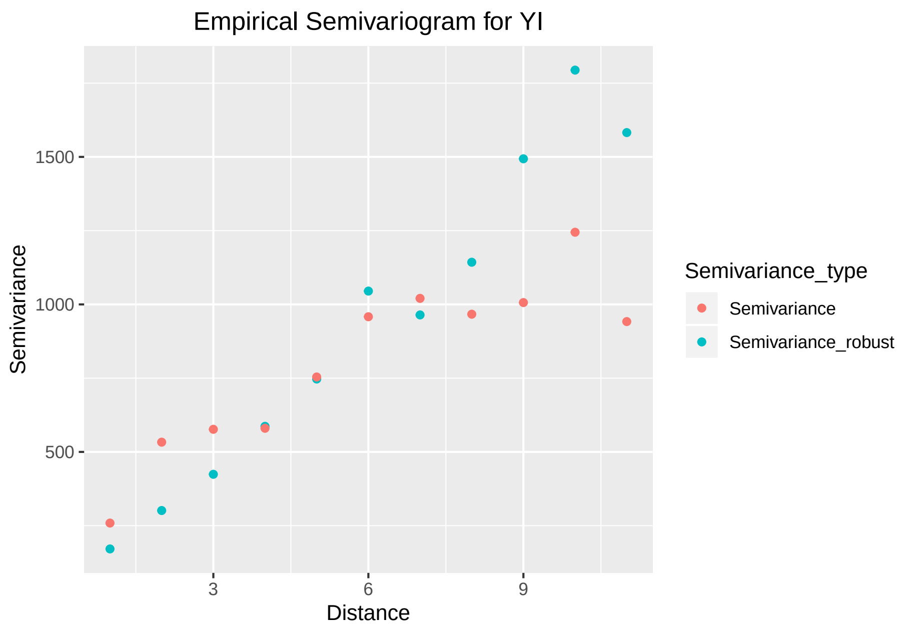
```
## variog: computing omnidirectional variogram
```

```
# extract the data
v1_robust_data <- cbind(v1_robust$u, v1_robust$v, v1_robust$n) %>%
  as.data.frame() %>%
  dplyr::rename(Distance = V1,
                Semivariance = V2,
                Pair_count = V3)

# plot robust variogram
```

```
v1_robust_vario <- ggplot(data = v1_robust_data) +
  geom_point(mapping = aes(x = Distance, y = Semivariance)) +
  ggtitle("Empirical Semivariogram of YI - Robust estimation") +
  theme(plot.title = element_text(hjust = 0.5))

v1_robust_vario
```



```
# combine robust and classical variogram
var_comb <- v1_robust_data %>%

  # combine robust and classical variogram datasets
  dplyr::rename(Semivariance_robust = Semivariance) %>%
  bind_cols(dplyr::select(v1_plot_data, Semivariance)) %>%
  gather(key = "Semivariance_type", value = "Semivariance", -c(Distance, Pair_count)) %>%

  # plot
  ggplot() +
  geom_point(mapping = aes(x = Distance, y = Semivariance, color = Semivariance_type)) +
  ggtitle("Empirical Semivariogram for YI") +
  theme(plot.title = element_text(hjust = 0.5))

var_comb
```



2.6 Variogram model selection

We will use the package `gstat` and `automap` for variogram model selection

```
# specify coordinates in the dataset
coordinates(a) = ~East+North

# select the best model out of exponential, spherical, and gaussian
autofitVariogram(YI ~ East + North, a, model = c("Sph", "Exp", "Gau"))
```

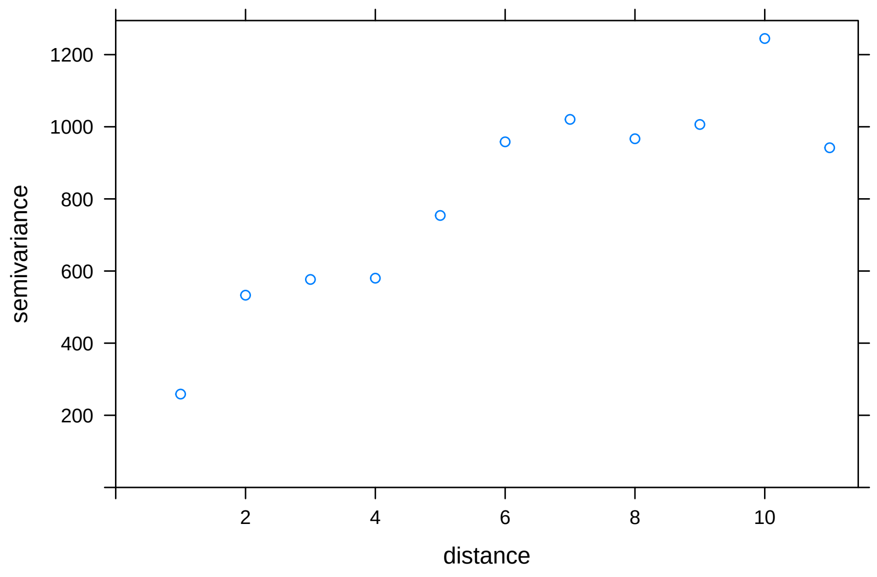
```
## $exp_var
##   np dist    gamma dir.hor dir.ver   id
## 1 15    1 258.8333       0       0 var1
## 2 14    2 533.0000       0       0 var1
## 3 13    3 576.6154       0       0 var1
## 4 12    4 580.1667       0       0 var1
## 5 11    5 754.0000       0       0 var1
##
## $var_model
##   model    psill    range
## 1   Nug   0.0000 0.000000
## 2   Exp 854.3133 2.575499
##
```

```
## $sserr
## [1] 28783.32
##
## attr(,"class")
## [1] "autofitVariogram" "list"
```

```
# fit empirical variogram
v_emp <- variogram(YI ~ East + North, data = a, cutoff = 11)
v_emp
```

```
##    np dist     gamma dir.hor dir.ver   id
## 1  15    1  258.8333       0       0 var1
## 2  14    2  533.0000       0       0 var1
## 3  13    3  576.6154       0       0 var1
## 4  12    4  580.1667       0       0 var1
## 5  11    5  754.0000       0       0 var1
## 6  10    6  958.2000       0       0 var1
## 7   9    7 1020.4444       0       0 var1
## 8   8    8  966.7500       0       0 var1
## 9   7    9 1006.2857       0       0 var1
## 10  6   10 1244.6667       0       0 var1
## 11  5   11  941.8000       0       0 var1
```

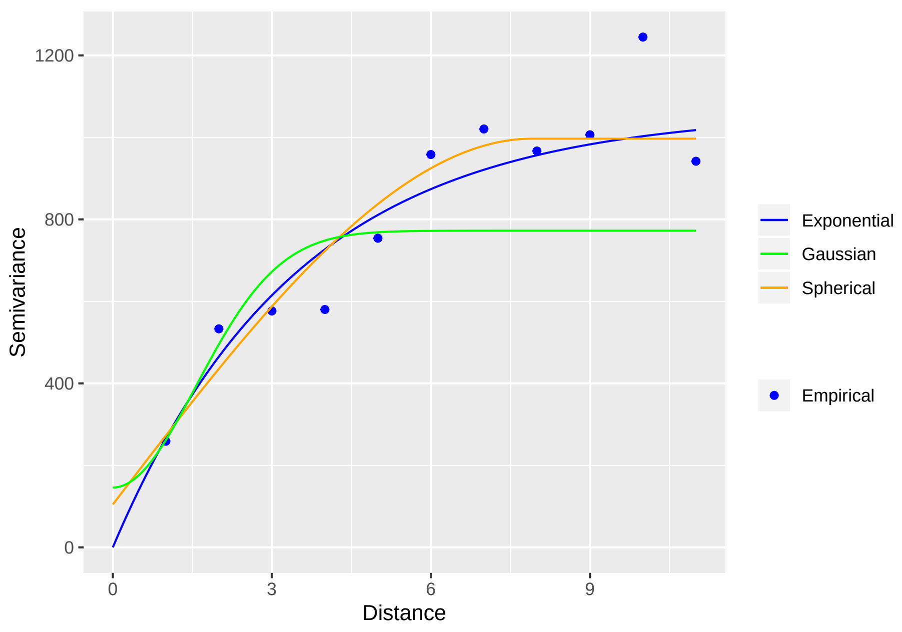
```
plot(v_emp)
```



```r
# fit exponential variogram
v_exp <- fit.variogram(v_emp, vgm("Exp"))
v_exp
```

```
##   model    psill   range
## 1   Nug    0.000 0.00000
## 2   Exp 1062.461 3.47171
```

```r
# fit spherical and gaussian
v_sph <- fit.variogram(v_emp, vgm("Sph"))
v_gau <- fit.variogram(v_emp, vgm("Gau"))

# extract plotting data from fitted variograms
v_exp_line <- variogramLine(v_exp, maxdist = 11)
v_sph_line <- variogramLine(v_sph, maxdist = 11)
v_gau_line <- variogramLine(v_gau, maxdist = 11)

# plot emprical and fitted variograms together
# specify color for legends
legend_color <- c("Empirical" = "blue", "Exponential" = "blue",
                  "Spherical" = "orange", "Gaussian" = "green")
ggplot(data = v_emp) +
  geom_point(mapping = aes(x = dist, y = gamma, fill = "Empirical"), color = "blue") +
  geom_line(data = v_exp_line, mapping = aes(x = dist, y = gamma, color = "Exponential
  geom_line(data = v_sph_line, mapping = aes(x = dist, y = gamma, color = "Spherical"))
  geom_line(data = v_gau_line, mapping = aes(x = dist, y = gamma, color = "Gaussian"))
  scale_color_manual(name = "", values = legend_color) +
  scale_fill_manual(name = "", values = legend_color) +
  labs(x = "Distance",
       y = "Semivariance")
```



Semivariance
1200
800
400
0
0
3
6
9
Distance
Exponential
Gaussian
Spherical
Empirical

Chapter 3

Exercise 9.5

3.1 Load packages

Here is the R code to download the required packages for this exercise.

```
# install package manager 'pacman'
if (!require(pacman)){
  install.packages('pacman')
}
```

```
## Loading required package: pacman
```

```
# load packages needed for this exercise
library(pacman)
p_load(tidyverse,
       lctools, # to calculate Moran's I
       spdep, # to calculate geary's c
       geoR, # to compute variogram
       gridExtra, # to stack plots
       gstat, automap, # packages for variogram model selection
       sp # need a function called `coordinates`
       )
```

3.2 Data

This is equivalent to data step in SAS. Here, the data is imported from a file `data.csv` using the function `read_csv`. This function will download the file directly from here.

```
# Import data
a <- read_csv("https://raw.githubusercontent.com/luckymehra/epidem-exercises/master/da
```

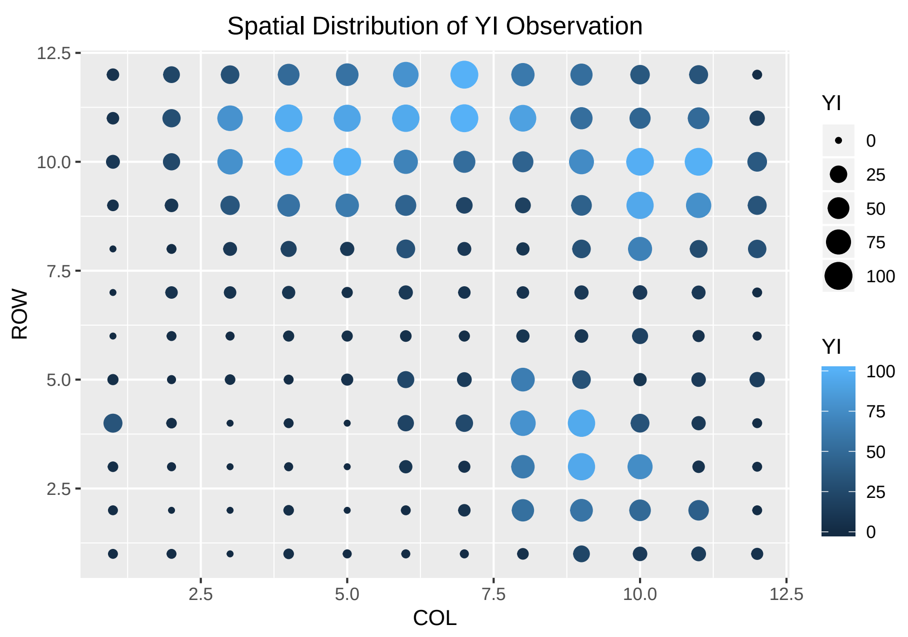
```
## Parsed with column specification:
## cols(
##   COL = col_double(),
##   ROW = col_double(),
##   YI = col_double()
## )
```

```
# print the data
a
```

```
## # A tibble: 144 x 3
##      COL   ROW    YI
##    <dbl> <dbl> <dbl>
##  1     1     1     2
##  2     2     1     2
##  3     3     1     0
##  4     4     1     3
##  5     5     1     1
##  6     6     1     1
##  7     7     1     1
##  8     8     1     5
##  9     9     1    22
## 10    10     1    13
## # ... with 134 more rows
```

3.3 Autocorrelation statistics

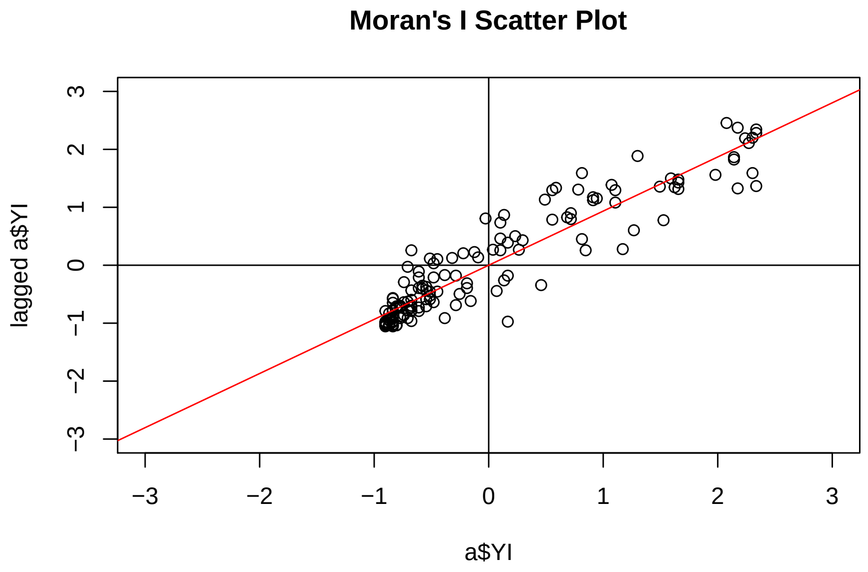
```
# visualize the data
ggplot(data = a) +
  geom_point(mapping = aes(x = COL, y = ROW, size = YI, color = YI)) +
  ggtitle("Spatial Distribution of YI Observation") +
  theme(plot.title = element_text(hjust = 0.5))
```



```
# calculate Moran's I
Coords <- a %>%
  dplyr::select(COL, ROW)

mI <- moransI(Coords, Bandwidth = 1, a$YI)

# print Moran's I table
moran.table <- tribble(
  ~`Moran's I`, ~`Expected I`, ~`Z randomization`, ~`P value randomization`,
  #------------/--------------/-------------------/------------------------
  mI$Morans.I, mI$Expected.I,  mI$z.randomization, mI$p.value.randomization
  )

moran.table
```

```
## # A tibble: 1 x 4
##   `Moran's I` `Expected I` `Z randomization` `P value randomization`
##         <dbl>        <dbl>             <dbl>                   <dbl>
## 1       0.782     -0.00699              13.0                1.28e-38
```

```
# create Moran's I scatter plot
l.moran <- l.moransI(Coords, Bandwidth = 1, a$YI)
```



```r
# calculate geary's c
Coords_num <- coordinates(Coords)

# create an object of class 'nb' so that it can be used with function from packege `sp
Coords_nb <- knn2nb(knearneigh(Coords_num))

# create a 'listw' object for use in the function `geary.test`
coords_listw <- nb2listw(Coords_nb)

gearyC <- geary.test(a$YI, coords_listw, alternative = "two.sided")
gearyC
```

```
##
##  Geary C test under randomisation
##
## data:  a$YI
## weights: coords_listw
##
## Geary C statistic standard deviate = 8.8657, p-value < 2.2e-16
## alternative hypothesis: two.sided
## sample estimates:
## Geary C statistic       Expectation          Variance
##       0.235058006       1.000000000       0.007444457
```

3.4 First variogram

We will use the package geoR to construct emprical variogram, and then draw them using package ggplot2.

```
v1 <- variog(coords = Coords_num, data = a$YI, breaks = seq(0.5, 15.5),
             max.dist = 12)
```

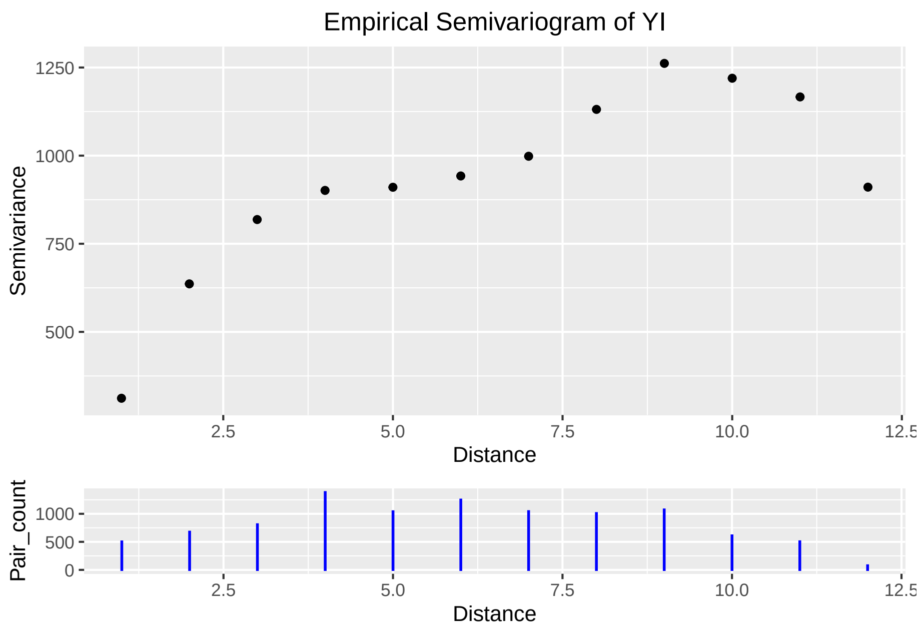
```
## variog: computing omnidirectional variogram
```

```
# extract data from object v1 for plotting
v1_plot_data <- cbind(v1$u, v1$v, v1$n) %>%
  as.data.frame() %>%
  dplyr::rename(Distance = V1,
                Semivariance = V2,
                Pair_count = V3)

# in the table below, gamma is semivariance
v1_plot_data
```

```
##    Distance Semivariance Pair_count
## 1         1     311.7154        506
## 2         2     636.2074        680
## 3         3     818.7044        812
## 4         4     901.3218       1386
## 5         5     910.2773       1044
## 6         6     942.3219       1252
## 7         7     998.2290       1046
## 8         8    1131.2105       1012
## 9         9    1261.6817       1076
## 10       10    1219.6067        614
## 11       11    1166.5541        508
## 12       12     910.6250         80
```

```
# plot variogram
v1_plot_vario <- ggplot(data = v1_plot_data) +
  geom_point(mapping = aes(x = Distance, y = Semivariance)) +
  ggtitle("Empirical Semivariogram of YI") +
  theme(plot.title = element_text(hjust = 0.5))

# plot pair counts
v1_plot_pair_count <- ggplot(data = v1_plot_data) +
  geom_col(mapping = aes(x = Distance, y = Pair_count), width = 0.01, color = "blue")
```

```
# stack two plots
grid.arrange(v1_plot_vario, v1_plot_pair_count,
             ncol = 1, heights = c(3, 1))
```



3.5 Second variogram

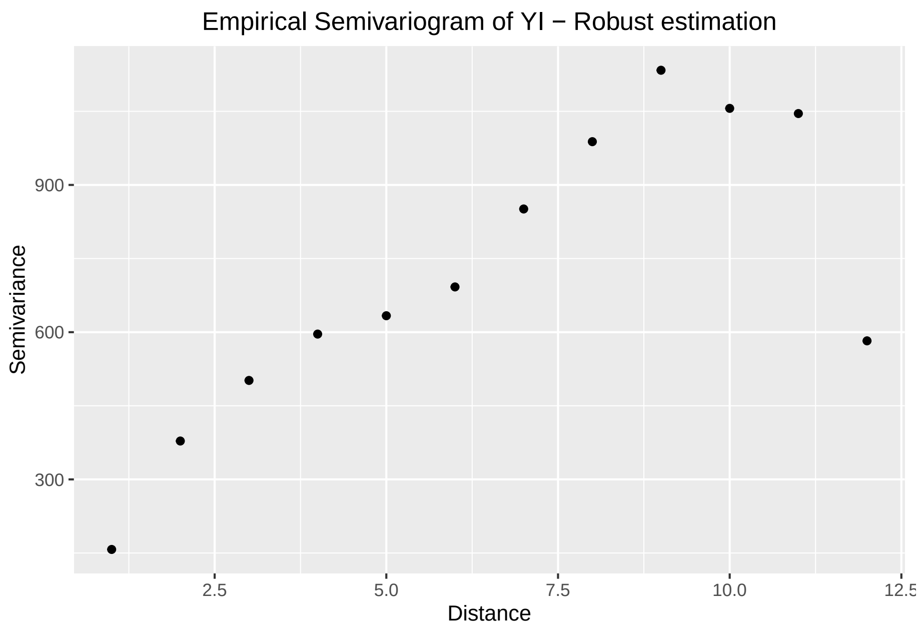
Plot robust and classical variogram together.

```
# fit robust variogram
v1_robust <- variog(coords = Coords_num, data = a$YI, breaks = seq(0.5, 15.5),
             max.dist = 12, estimator.type = "modulus")
```

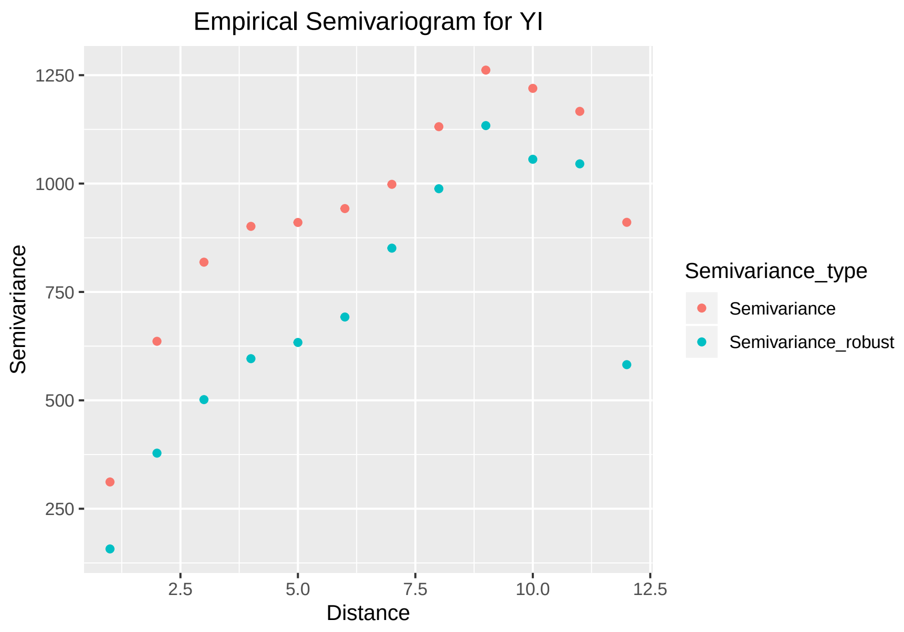
```
## variog: computing omnidirectional variogram
```

```
# extract the data
v1_robust_data <- cbind(v1_robust$u, v1_robust$v, v1_robust$n) %>%
  as.data.frame() %>%
  dplyr::rename(Distance = V1,
                Semivariance = V2,
                Pair_count = V3)
```

```
# plot robust variogram
v1_robust_vario <- ggplot(data = v1_robust_data) +
  geom_point(mapping = aes(x = Distance, y = Semivariance)) +
  ggtitle("Empirical Semivariogram of YI - Robust estimation") +
  theme(plot.title = element_text(hjust = 0.5))

v1_robust_vario
```



```
# combine robust and classical variogram
var_comb <- v1_robust_data %>%

  # combine robust and classical variogram datasets
  dplyr::rename(Semivariance_robust = Semivariance) %>%
  bind_cols(dplyr::select(v1_plot_data, Semivariance)) %>%
  gather(key = "Semivariance_type", value = "Semivariance", -c(Distance, Pair_count)) %>%

  # plot
  ggplot() +
  geom_point(mapping = aes(x = Distance, y = Semivariance, color = Semivariance_type)) +
  ggtitle("Empirical Semivariogram for YI") +
  theme(plot.title = element_text(hjust = 0.5))

var_comb
```



3.6 Variogram model selection

We will use the package `gstat` and `automap` for variogram model selection

```
# specify coordinates in the dataset
coordinates(a) = ~COL+ROW

# select the best model out of exponential, spherical, and gaussian
autofitVariogram(YI ~ COL + ROW, a, model = c("Sph", "Exp", "Gau"), cutoff = 12)
```

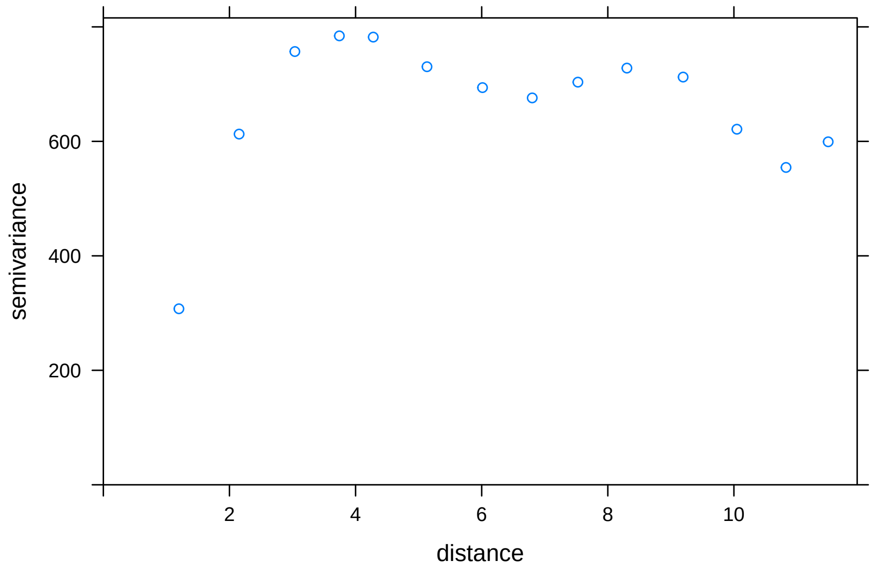
```
## $exp_var
##     np     dist    gamma dir.hor dir.ver   id
## 1  264 1.000000 233.2400       0       0 var1
## 2  242 1.414214 388.5222       0       0 var1
## 3  680 2.152750 612.6985       0       0 var1
## 4  812 3.036881 756.8971       0       0 var1
## 5 1066 3.944315 783.1461       0       0 var1
## 6 1364 4.977586 742.6252       0       0 var1
##
## $var_model
##   model    psill    range
## 1   Nug   0.0000 0.000000
## 2   Sph 782.9935 4.019145
```

```
##
## $sserr
## [1] 1247749
##
## attr(,"class")
## [1] "autofitVariogram" "list"
```

```
# fit empirical variogram
v_emp <- variogram(YI ~ COL + ROW, data = a, cutoff = 12)
v_emp
```

```
##      np      dist    gamma dir.hor dir.ver   id
## 1   506  1.198102 307.5054       0       0 var1
## 2   680  2.152750 612.6985       0       0 var1
## 3   812  3.036881 756.8971       0       0 var1
## 4   552  3.742751 784.3027       0       0 var1
## 5   834  4.280245 782.1560       0       0 var1
## 6  1044  5.132514 730.3844       0       0 var1
## 7  1028  6.012860 693.9058       0       0 var1
## 8   878  6.801676 675.9157       0       0 var1
## 9   836  7.525735 703.4337       0       0 var1
## 10  852  8.302717 728.0099       0       0 var1
## 11  792  9.194510 712.3311       0       0 var1
## 12  542 10.047104 621.2100       0       0 var1
## 13  452 10.826377 554.3985       0       0 var1
## 14  208 11.494850 599.2237       0       0 var1
```

```
plot(v_emp)
```



```r
# fit exponential variogram
v_exp <- fit.variogram(v_emp, vgm("Exp"))

# fit spherical and gaussian
v_sph <- fit.variogram(v_emp, vgm("Sph"))
v_sph
```

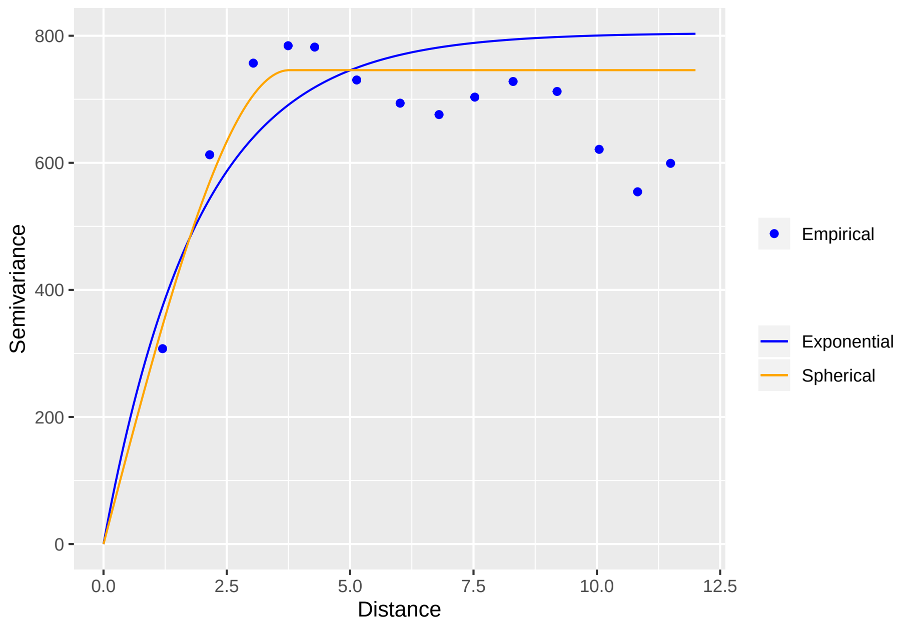
```
##   model    psill    range
## 1   Nug   0.0000 0.000000
## 2   Sph 745.8602 3.765221
```

```r
v_gau <- fit.variogram(v_emp, vgm("Gau"))
```

```
## Warning in fit.variogram(v_emp, vgm("Gau")): No convergence after 200
## iterations: try different initial values?
```

```r
# extract plotting data from fitted variograms
v_exp_line <- variogramLine(v_exp, maxdist = 12)
v_sph_line <- variogramLine(v_sph, maxdist = 12)
# v_gau_line <- variogramLine(v_gau, maxdist = 12)

# plot emprical and fitted variograms together
# specify color for legends
```

```
legend_color <- c("Empirical" = "blue", "Exponential" = "blue",
                  "Spherical" = "orange")
ggplot(data = v_emp) +
  geom_point(mapping = aes(x = dist, y = gamma, fill = "Empirical"), color = "blue") +
  geom_line(data = v_exp_line, mapping = aes(x = dist, y = gamma, color = "Exponential")) +
  geom_line(data = v_sph_line, mapping = aes(x = dist, y = gamma, color = "Spherical")) +
  # geom_line(data = v_gau_line, mapping = aes(x = dist, y = gamma, color = "Gaussian")) +
  scale_color_manual(name = "", values = legend_color) +
  scale_fill_manual(name = "", values = legend_color) +
  labs(x = "Distance",
       y = "Semivariance")
```



Chapter 4

Yield loss

4.1 Load packages

Here is the R code to download the required packages for this exercise.

```
# install package manager 'pacman'
if (!require(pacman)){
  install.packages('pacman')
}
```

```
## Loading required package: pacman
```

```
# load packages needed for this exercise
pacman::p_load(tidyverse,
       nlme,
       emmeans,
       predictmeans
       )
```

4.2 Data

This is equivalent to the data step in SAS. Here, the data is imported from a file `yield_loss.csv` using the function `read_csv`. This function will download the data file direclty from here.

```
# Import data
a <- read_csv("https://raw.githubusercontent.com/luckymehra/epidem-exercises/master/data/yield_lo
## Parsed with column specification:
## cols(
##   WP = col_double(),
##   SP = col_character(),
##   BLK = col_double(),
##   TRT = col_double(),
##   FUNG = col_double(),
##   DS = col_double(),
##   YIELD = col_double()
## )
```

```
# print the data
a
```

```
## # A tibble: 24 x 7
##        WP SP      BLK   TRT  FUNG    DS YIELD
##     <dbl> <chr> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1   101 A         1     1     0    43   205
##  2   101 B         1     1     1     1   399
##  3   102 A         1     2     1     2   426
##  4   102 B         1     2     0    92   102
##  5   103 A         1     3     1     2   385
##  6   103 B         1     3     0     7   355
##  7   201 A         2     2     1     4   412
##  8   201 B         2     2     0    75   224
##  9   202 A         2     3     1     3   425
## 10   202 B         2     3     0    10   352
## # ... with 14 more rows
```

```
# specify that FUNG, TRT, and BLK are factors
a$FUNG <- as.ordered(as.factor(a$FUNG))
a$TRT <- as.ordered(as.factor(a$TRT))
a$BLK <- as.ordered(as.factor(a$BLK))
```

4.3 Mixed model for response variable DS

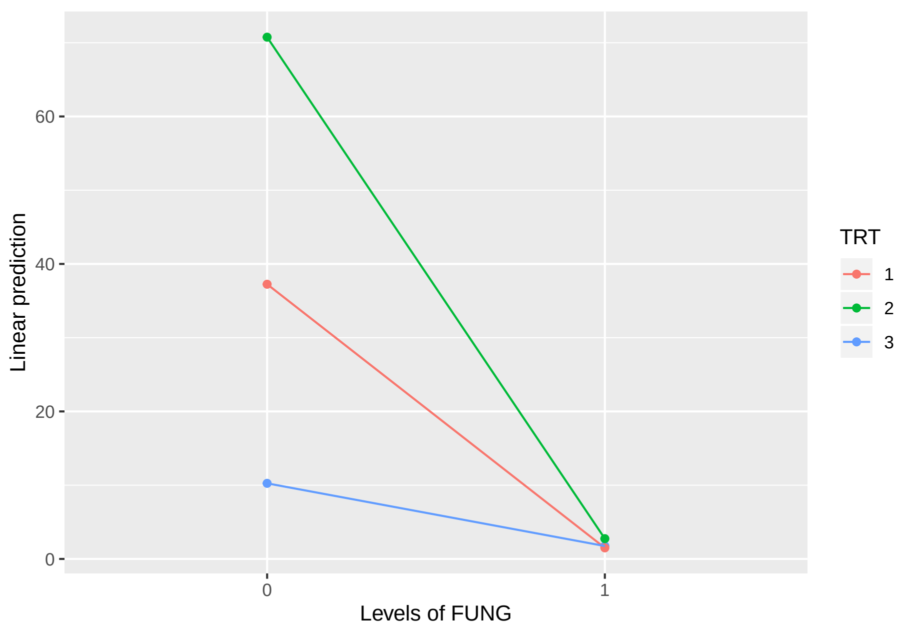
```
# fit the model
mm_1 <- lme(DS ~ TRT*FUNG, # fixed effects
            data = a,
            random = ~1|BLK/TRT) # read mm_1 as mixed model 1

# summary output
summary(mm_1)
```

```
## Linear mixed-effects model fit by REML
##  Data: a
##        AIC      BIC    logLik
##   156.1691 164.1825 -69.08456
##
## Random effects:
##  Formula: ~1 | BLK
##          (Intercept)
## StdDev: 0.0009561632
##
##  Formula: ~1 | TRT %in% BLK
##         (Intercept) Residual
## StdDev: 0.001007113 7.918859
##
## Fixed effects: DS ~ TRT * FUNG
##                    Value Std.Error DF    t-value p-value
## (Intercept)    20.708333  1.616430  9  12.811150  0.0000
## TRT.L          -9.457553  2.799740  6  -3.378012  0.0149
## TRT.Q         -19.646949  2.799740  6  -7.017420  0.0004
## FUNG.L        -26.457579  2.285978  9 -11.573857  0.0000
## TRT.L:FUNG.L   13.625000  3.959430  9   3.441152  0.0074
## TRT.Q:FUNG.L   26.485944  3.959430  9   6.689333  0.0001
##  Correlation:
##              (Intr) TRT.L TRT.Q FUNG.L TRT.L:
## TRT.L        0
## TRT.Q        0      0
## FUNG.L       0      0     0
## TRT.L:FUNG.L 0      0     0     0
## TRT.Q:FUNG.L 0      0     0     0      0
##
## Standardized Within-Group Residuals:
##           Min            Q1           Med            Q3           Max
## -2.241484e+00 -9.471064e-02 -1.590814e-08  1.736361e-01  2.683467e+00
##
## Number of Observations: 24
## Number of Groups:
##          BLK TRT %in% BLK
##            4           12
```

```
# type 3 tests of fixed effects
anova(mm_1)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1     9 164.12557  <.0001
## TRT             2     6  30.32757   7e-04
## FUNG            1     9 133.95416  <.0001
## TRT:FUNG        2     9  28.29435   1e-04
```

```
# visualize interaction
emmip(mm_1, TRT ~ FUNG)
```



```
# to do anova for random effects, we need to compare mm_1 with a model that only has f
# we can use `gls()` function in `nlme` to fit the fixed effects model
fixed_model <- gls(DS ~ TRT * FUNG,
                                      data = a)

# test the random effects in the model
anova(mm_1, fixed_model)
```

```
##             Model df      AIC      BIC    logLik   Test      L.Ratio
## mm_1            1  9 156.1691 164.1825 -69.08456
## fixed_model     2  7 152.1691 158.4017 -69.08456 1 vs 2 1.250038e-08
##             p-value
## mm_1
## fixed_model       1
```

```
# least square means
test(emmeans(mm_1, "TRT"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  TRT emmean  SE df t.ratio p.value
##  1     19.4 2.8  3  6.920  0.0062
##  2     36.8 2.8  3 13.126  0.0010
##  3      6.0 2.8  3  2.143  0.1215
##
## Results are averaged over the levels of: FUNG
## d.f. method: containment
```

```
test(emmeans(mm_1, "FUNG"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  FUNG emmean   SE df t.ratio p.value
##  0      39.4 2.29  3 17.243  0.0004
##  1       2.0 2.29  3  0.875  0.4460
##
## Results are averaged over the levels of: TRT
## d.f. method: containment
```

```
# pairwise difference
test(emmeans(mm_1, pairwise ~ TRT), adjust = "none")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
## $emmeans
##  TRT emmean  SE df t.ratio p.value
##  1     19.4 2.8  3  6.920  0.0062
##  2     36.8 2.8  3 13.126  0.0010
##  3      6.0 2.8  3  2.143  0.1215
##
## Results are averaged over the levels of: FUNG
## d.f. method: containment
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  1 - 2       -17.4 3.96  6 -4.388  0.0046
##  1 - 3        13.4 3.96  6  3.378  0.0149
##  2 - 3        30.8 3.96  6  7.766  0.0002
##
## Results are averaged over the levels of: FUNG
```

```r
test(emmeans(mm_1, pairwise ~ FUNG))
```

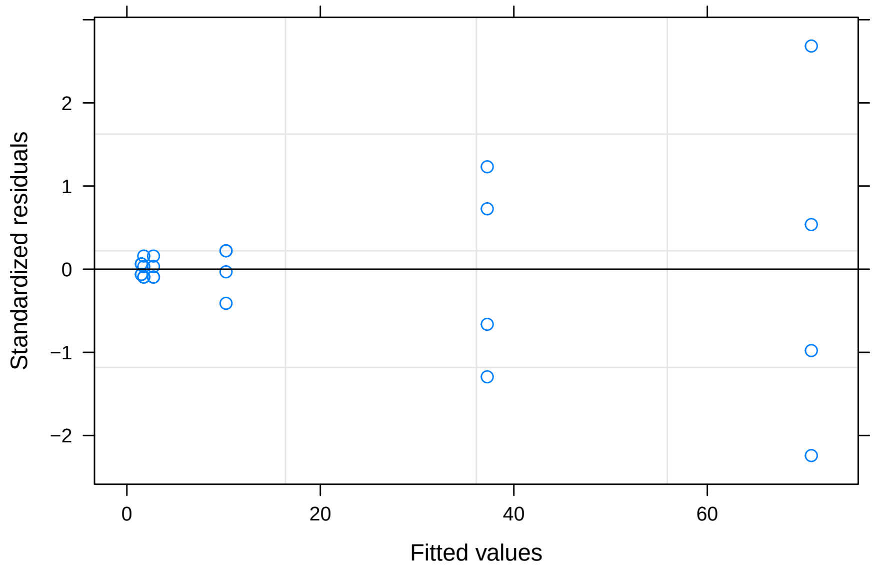
```
## NOTE: Results may be misleading due to involvement in interactions

## $emmeans
##  FUNG emmean   SE df t.ratio p.value
##  0      39.4 2.29  3 17.243  0.0004
##  1       2.0 2.29  3  0.875  0.4460
##
## Results are averaged over the levels of: TRT
## d.f. method: containment
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  0 - 1        37.4 3.23  9 11.574  <.0001
##
## Results are averaged over the levels of: TRT
```

```r
test(emmeans(mm_1, pairwise ~ TRT*FUNG), adjust = "none")
```

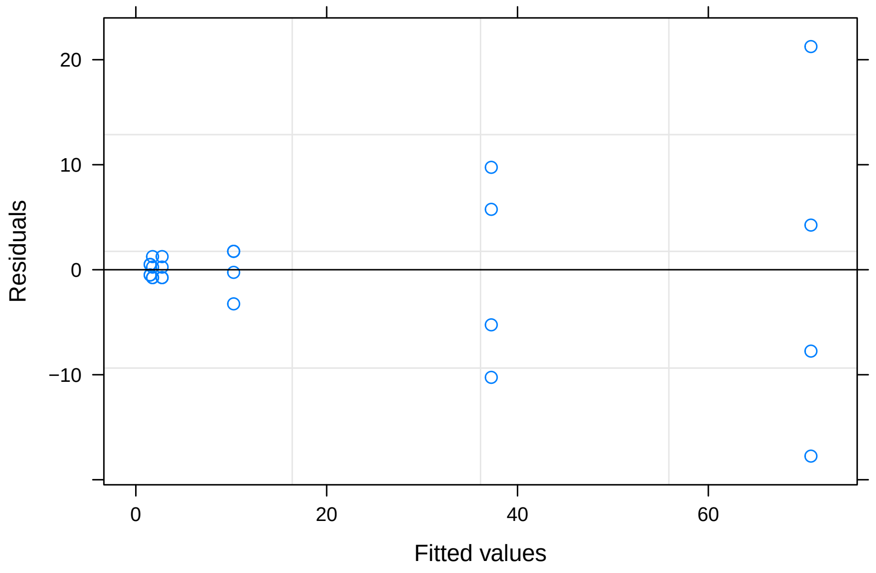
```
## $emmeans
##  TRT FUNG emmean   SE df t.ratio p.value
##  1   0     37.25 3.96  3  9.408  0.0025
##  2   0     70.75 3.96  3 17.869  0.0004
##  3   0     10.25 3.96  3  2.589  0.0812
##  1   1      1.50 3.96  3  0.379  0.7300
##  2   1      2.75 3.96  3  0.695  0.5373
##  3   1      1.75 3.96  3  0.442  0.6884
##
## d.f. method: containment
##
## $contrasts
##  contrast  estimate  SE df t.ratio p.value
##  1,0 - 2,0   -33.50 5.6  6 -5.983  0.0010
##  1,0 - 3,0    27.00 5.6  6  4.822  0.0029
##  1,0 - 1,1    35.75 5.6  9  6.385  0.0001
##  1,0 - 2,1    34.50 5.6  6  6.161  0.0008
##  1,0 - 3,1    35.50 5.6  6  6.340  0.0007
##  2,0 - 3,0    60.50 5.6  6 10.805  <.0001
##  2,0 - 1,1    69.25 5.6  6 12.367  <.0001
##  2,0 - 2,1    68.00 5.6  9 12.144  <.0001
##  2,0 - 3,1    69.00 5.6  6 12.323  <.0001
##  3,0 - 1,1     8.75 5.6  6  1.563  0.1692
##  3,0 - 2,1     7.50 5.6  6  1.339  0.2289
##  3,0 - 3,1     8.50 5.6  9  1.518  0.1633
##  1,1 - 2,1    -1.25 5.6  6 -0.223  0.8308
##  1,1 - 3,1    -0.25 5.6  6 -0.045  0.9658
##  2,1 - 3,1     1.00 5.6  6  0.179  0.8641
```

4.3.1 Diagnostic plots

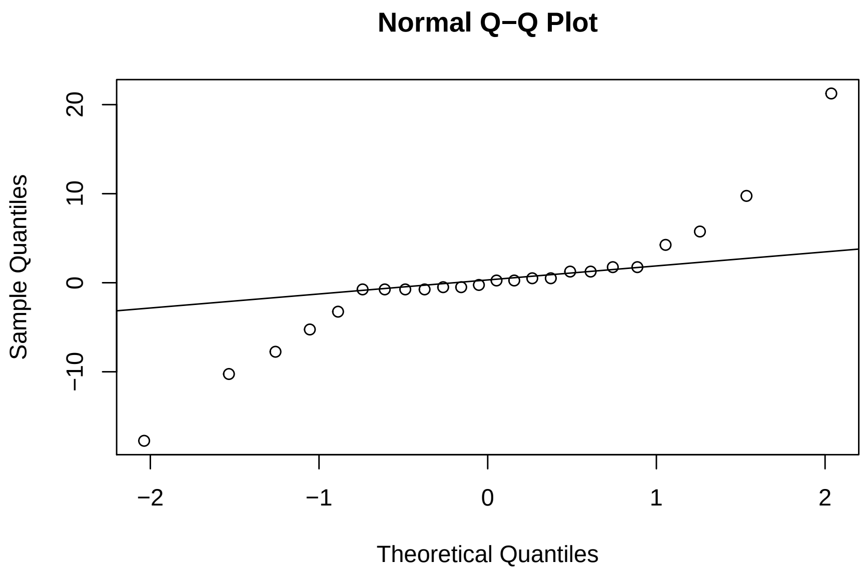
```r
# pearson residuals vs. fitted values
plot(mm_1, resid(., type="pearson") ~ fitted(.), abline = 0)
```



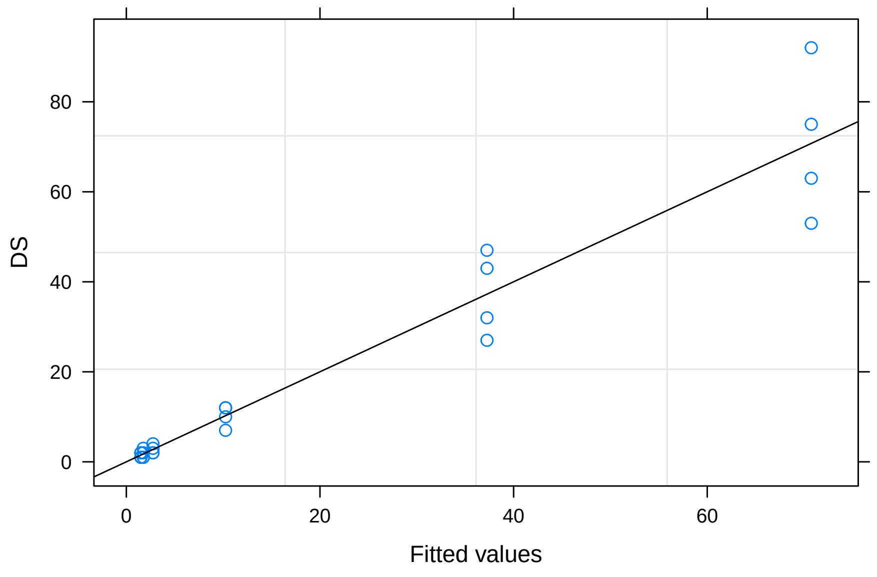
```r
# standardaized residuals vs. fitted values
plot(mm_1, resid(., scaled=TRUE) ~ fitted(.), abline = 0)
```



```r
# qq plot
qqnorm(residuals(mm_1))
qqline(residuals(mm_1))
```



```
#observed vs. fitted values
plot(mm_1, DS ~ fitted(.), abline = c(0,1))
```



4.4 Mixed model for response variable YIELD

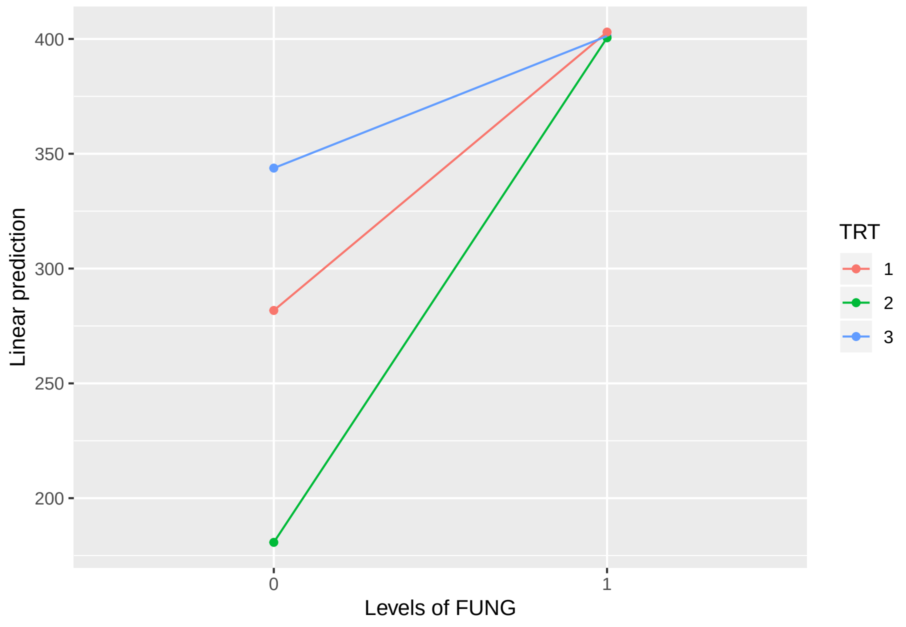
```
# fit the model
mm_2 <- lme(YIELD ~ TRT*FUNG, # fixed effects
            data = a,
            random = ~1|BLK/TRT) # read mm_2 as mixed model 2

# summary output
summary(mm_2)
```

```
## Linear mixed-effects model fit by REML
##  Data: a
##         AIC      BIC   logLik
##    209.9214 217.9348 -95.9607
##
## Random effects:
##  Formula: ~1 | BLK
##          (Intercept)
## StdDev:    11.99815
##
##  Formula: ~1 | TRT %in% BLK
##         (Intercept) Residual
## StdDev: 0.001914579 33.61779
##
## Fixed effects: YIELD ~ TRT * FUNG
##                  Value Std.Error DF  t-value p-value
## (Intercept)  335.1667  9.114752  9 36.77189  0.0000
## TRT.L         21.3016 11.885682  6  1.79221  0.1233
## TRT.Q         54.5522 11.885682  6  4.58974  0.0037
## FUNG.L        93.9274  9.704619  9  9.67862  0.0000
## TRT.L:FUNG.L -31.8750 16.808893  9 -1.89632  0.0904
## TRT.Q:FUNG.L -75.2720 16.808893  9 -4.47811  0.0015
##  Correlation:
##              (Intr) TRT.L TRT.Q FUNG.L TRT.L:
## TRT.L        0
## TRT.Q        0      0
## FUNG.L       0      0     0
## TRT.L:FUNG.L 0      0     0     0
## TRT.Q:FUNG.L 0      0     0     0      0
##
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max
## -2.04399017 -0.33265713  0.07191314  0.49972251  1.20545632
##
## Number of Observations: 24
## Number of Groups:
##          BLK TRT %in% BLK
##            4           12
```

```
# type 3 tests of fixed effects
anova(mm_2)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1     9 1352.1720  <.0001
## TRT             2     6   12.1389  0.0078
## FUNG            1     9   93.6757  <.0001
## TRT:FUNG        2     9   11.8247  0.0030
```

```
# visualize interaction
emmip(mm_2, TRT ~ FUNG)
```



```r
# to do anova for random effects, we need to compare mm_1 with a model that only has fixed effec
# we can use `gls()` function in `nlme` to fit the fixed effects model
fixed_model_YIELD <- gls(YIELD ~ TRT * FUNG,
                                     data = a)

# test the random effects in the model
anova(mm_2, fixed_model_YIELD)
```

```
##                   Model df      AIC      BIC    logLik   Test   L.Ratio
## mm_2                  1  9 209.9214 217.9348 -95.96070
## fixed_model_YIELD     2  7 206.3763 212.6089 -96.18815 1 vs 2 0.4548877
##                   p-value
## mm_2
## fixed_model_YIELD  0.7966
```

```r
# least square means
test(emmeans(mm_2, "TRT"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  TRT emmean   SE df t.ratio p.value
##  1      342 13.3  3 25.716  0.0001
##  2      291 13.3  3 21.829  0.0002
##  3      372 13.3  3 27.978  0.0001
##
## Results are averaged over the levels of: FUNG
## d.f. method: containment
```

```r
test(emmeans(mm_2, "FUNG"))
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
##  FUNG emmean   SE df t.ratio p.value
##  0       269 11.4  3 23.556  0.0002
##  1       402 11.4  3 35.198  0.0001
##
## Results are averaged over the levels of: TRT
## d.f. method: containment
```

```r
# pairwise difference
test(emmeans(mm_2, pairwise ~ TRT), adjust = "none")
```

```
## NOTE: Results may be misleading due to involvement in interactions
```

```
## $emmeans
##  TRT emmean   SE df t.ratio p.value
##  1      342 13.3  3 25.716  0.0001
##  2      291 13.3  3 21.829  0.0002
##  3      372 13.3  3 27.978  0.0001
##
## Results are averaged over the levels of: FUNG
## d.f. method: containment
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  1 - 2        51.8 16.8  6   3.079  0.0217
##  1 - 3       -30.1 16.8  6  -1.792  0.1233
##  2 - 3       -81.9 16.8  6  -4.871  0.0028
##
## Results are averaged over the levels of: FUNG
```

```r
test(emmeans(mm_2, pairwise ~ FUNG))
```

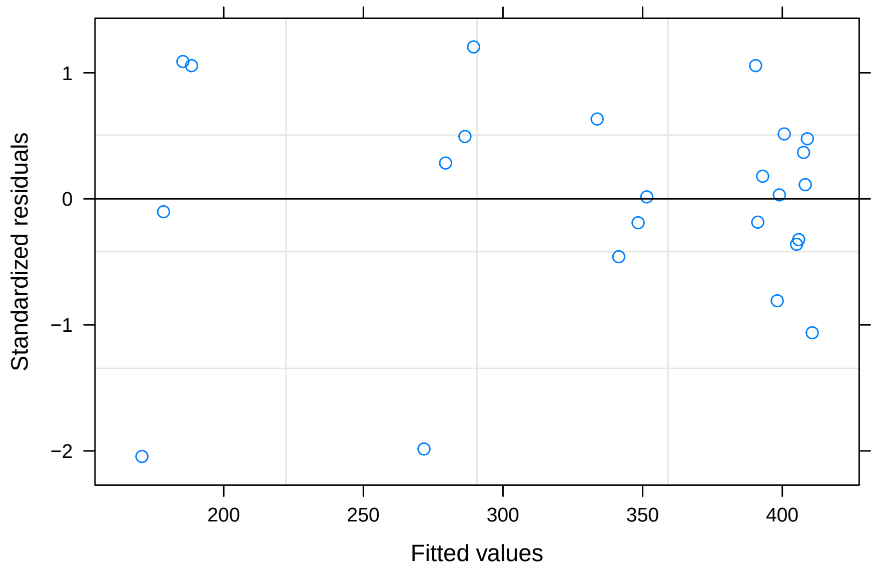
```
## NOTE: Results may be misleading due to involvement in interactions
## $emmeans
##  FUNG emmean   SE df t.ratio p.value
##  0       269 11.4  3 23.556  0.0002
##  1       402 11.4  3 35.198  0.0001
##
## Results are averaged over the levels of: TRT
## d.f. method: containment
##
## $contrasts
##  contrast estimate   SE df t.ratio p.value
##  0 - 1        -133 13.7  9 -9.679  <.0001
##
## Results are averaged over the levels of: TRT
```

```
test(emmeans(mm_2, pairwise ~ TRT*FUNG), adjust = "none")
```

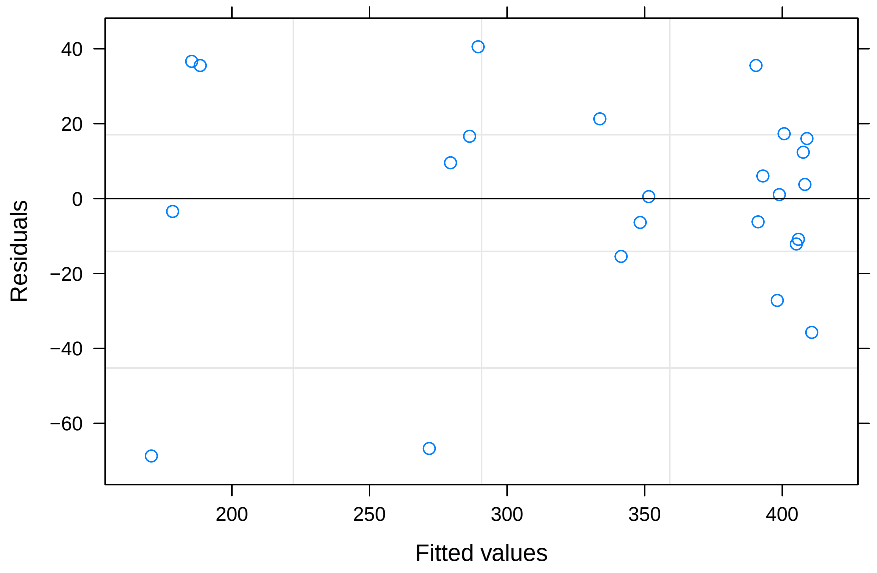
```
## $emmeans
##  TRT FUNG emmean   SE df t.ratio p.value
##  1   0       282 17.8  3 15.787  0.0006
##  2   0       181 17.8  3 10.128  0.0021
##  3   0       344 17.8  3 19.261  0.0003
##  1   1       403 17.8  3 22.580  0.0002
##  2   1       400 17.8  3 22.440  0.0002
##  3   1       401 17.8  3 22.482  0.0002
##
## d.f. method: containment
##
## $contrasts
##  contrast  estimate   SE df t.ratio p.value
##  1,0 - 2,0   101.00 23.8  6  4.249  0.0054
##  1,0 - 3,0   -62.00 23.8  6 -2.608  0.0402
##  1,0 - 1,1  -121.25 23.8  9 -5.101  0.0006
##  1,0 - 2,1  -118.75 23.8  6 -4.996  0.0025
##  1,0 - 3,1  -119.50 23.8  6 -5.027  0.0024
##  2,0 - 3,0  -163.00 23.8  6 -6.857  0.0005
##  2,0 - 1,1  -222.25 23.8  6 -9.349  0.0001
##  2,0 - 2,1  -219.75 23.8  9 -9.244  <.0001
##  2,0 - 3,1  -220.50 23.8  6 -9.276  0.0001
##  3,0 - 1,1   -59.25 23.8  6 -2.492  0.0470
##  3,0 - 2,1   -56.75 23.8  6 -2.387  0.0542
##  3,0 - 3,1   -57.50 23.8  9 -2.419  0.0387
##  1,1 - 2,1     2.50 23.8  6  0.105  0.9197
##  1,1 - 3,1     1.75 23.8  6  0.074  0.9437
##  2,1 - 3,1    -0.75 23.8  6 -0.032  0.9759
```

4.4.1 Diagnostic plots

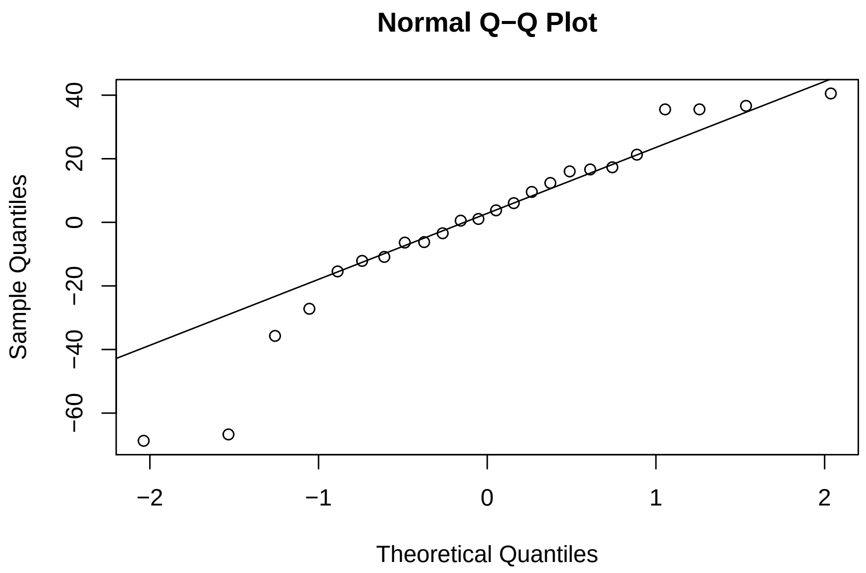
```r
# pearson residuals vs. fitted values
plot(mm_2, resid(., type="pearson") ~ fitted(.), abline = 0)
```



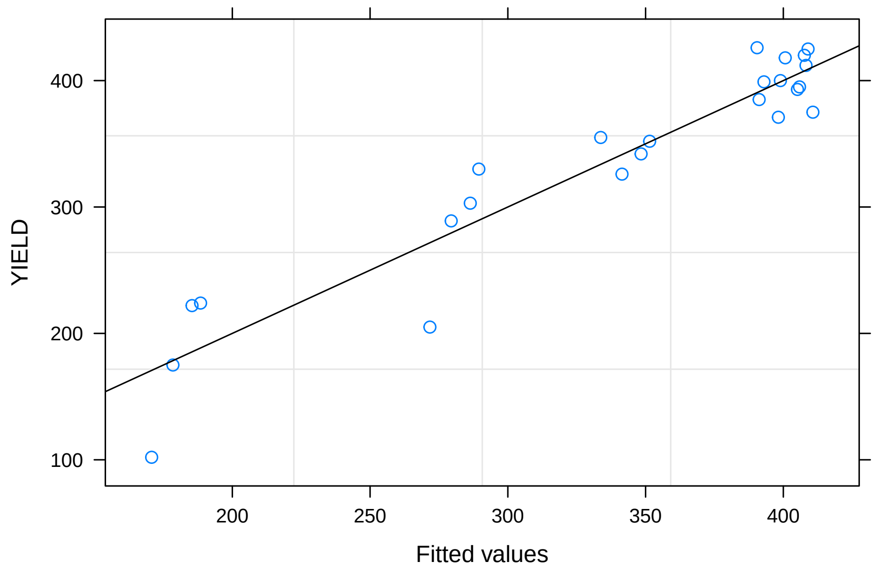
```r
# standardaized residuals vs. fitted values
plot(mm_2, resid(., scaled=TRUE) ~ fitted(.), abline = 0)
```



```
# qq plot
qqnorm(residuals(mm_2))
qqline(residuals(mm_2))
```



```
#observed vs. fitted values
plot(mm_2, YIELD ~ fitted(.), abline = c(0,1))
```



4.5 Linear regression between YIELD and DS

```
# fit `lm` model
lm_1 <- lm(YIELD ~ DS, data = a)
summary(lm_1)
```

```
##
## Call:
## lm(formula = YIELD ~ DS, data = a)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -61.196 -18.565   0.856  22.676  56.812
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 399.2384     8.0711   49.47  < 2e-16 ***
## DS           -3.0940     0.2399  -12.90 9.81e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.17 on 22 degrees of freedom
## Multiple R-squared:  0.8832, Adjusted R-squared:  0.8779
## F-statistic: 166.4 on 1 and 22 DF,  p-value: 9.809e-12
```

```
anova(lm_1)
```

```
## Analysis of Variance Table
##
## Response: YIELD
##           Df Sum Sq Mean Sq F value    Pr(>F)
## DS         1 161600  161600  166.38 9.809e-12 ***
## Residuals 22  21368     971
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# diagnostic plots
residplot(lm_1)
```

4.5.1 Linear regression between RY1 and DS

```
b <- a %>%
    mutate(RY1 = YIELD/399.23843)

# fit linear regression model
lm_2 <- lm(RY1 ~ DS, data = b)
summary(lm_2)
```

```
##
## Call:
## lm(formula = RY1 ~ DS, data = b)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.153282 -0.046502  0.002143  0.056798  0.142301
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.0000000  0.0202162   49.47  < 2e-16 ***
## DS          -0.0077498  0.0006008  -12.90 9.81e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07806 on 22 degrees of freedom
## Multiple R-squared:  0.8832, Adjusted R-squared:  0.8779
## F-statistic: 166.4 on 1 and 22 DF,  p-value: 9.809e-12
```

```
anova(lm_2)
```

```
## Analysis of Variance Table
##
## Response: RY1
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## DS         1 1.01385 1.01385  166.38 9.809e-12 ***
## Residuals 22 0.13406 0.00609
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# diagnostic plots
residplot(lm_2)
```

4.5.2 Transform dataset a

```
a_yield <- a %>%
    dplyr::select(BLK, TRT, YIELD) %>%
    arrange(BLK, TRT, YIELD) %>%
    group_by(BLK, TRT) %>%
    summarise(RY2 = YIELD[1]/YIELD[2]) %>%
    ungroup()


a_ds <- a %>%
    dplyr::select(BLK, TRT, DS) %>%
    arrange(BLK, TRT, DS) %>%
    group_by(BLK, TRT) %>%
    summarise(CDS = DS[2]) %>%
    ungroup()

a_new <- a_yield %>%
    inner_join(a_ds) %>%
    ungroup() %>%
    mutate(BLK = parse_factor(as.character(BLK)),
                 TRT = parse_factor(as.character(TRT)))
```

```
## Joining, by = c("BLK", "TRT")
```

```
# print the data
a_new
```

```
## # A tibble: 12 x 4
##    BLK   TRT     RY2   CDS
##    <fct> <fct> <dbl> <dbl>
##  1 1     1     0.514    43
##  2 1     2     0.239    92
##  3 1     3     0.922     7
##  4 2     1     0.88     27
##  5 2     2     0.544    75
##  6 2     3     0.828    10
##  7 3     1     0.721    47
##  8 3     2     0.565    63
##  9 3     3     0.866    12
## 10 4     1     0.691    32
## 11 4     2     0.472    53
## 12 4     3     0.815    12
```

4.6 Mixed model for RY2

```
# fit the model
mm_3 <- lme(RY2 ~ TRT, # fixed effects
            data = a_new,
            random = ~1|BLK) # read mm_3 as mixed model 3

# summary output
summary(mm_3)
```

```
## Linear mixed-effects model fit by REML
##  Data: a_new
##       AIC      BIC   logLik
##   2.04077 3.026893 3.979615
##
## Random effects:
##  Formula: ~1 | BLK
##         (Intercept)  Residual
## StdDev:  0.05297273 0.1134963
##
## Fixed effects: RY2 ~ TRT
##                  Value  Std.Error DF   t-value p-value
## (Intercept)  0.7016501 0.06262490  6 11.204012  0.0000
## TRT2        -0.2467228 0.08025398  6 -3.074274  0.0218
## TRT3         0.1561339 0.08025398  6  1.945497  0.0997
##  Correlation: 
##      (Intr) TRT2 
## TRT2 -0.641      
## TRT3 -0.641  0.500
## 
## Standardized Within-Group Residuals:
##         Min          Q1         Med          Q3         Max 
## -1.50508335 -0.38516867 -0.01698779  0.58195830  1.29565814 
## 
## Number of Observations: 12
## Number of Groups: 4
```

```r
# type 3 tests of fixed effects
anova(mm_3)
```

```
##             numDF denDF   F-value p-value
## (Intercept)     1     6 254.00333  <.0001
## TRT             2     6  12.81141  0.0068
```

```r
# to do anova for random effects, we need to compare mm_1 with a model that only has f
# we can use `gls()` function in `nlme` to fit the fixed effects model
fixed_model_RY2 <- gls(RY2 ~ TRT,
                                    data = a_new)

# test the random effects in the model
anova(mm_3, fixed_model_RY2)
```

```
##                 Model df       AIC      BIC   logLik   Test   L.Ratio
## mm_3                1  5 2.0407705 3.026893 3.979615                 
## fixed_model_RY2     2  4 0.3057644 1.094663 3.847118 1 vs 2 0.2649939
##                 p-value
## mm_3                   
## fixed_model_RY2  0.6067
```

```r
# pairwise difference
test(emmeans(mm_3, pairwise ~ TRT), adjust = "none")
```

```
## $emmeans
##  TRT emmean     SE df t.ratio p.value
##  1    0.702 0.0626  3 11.204  0.0015
##  2    0.455 0.0626  3  7.264  0.0054
##  3    0.858 0.0626  3 13.697  0.0008
##
## d.f. method: containment
##
## $contrasts
##  contrast estimate     SE df t.ratio p.value
##  1 - 2       0.247 0.0803  6   3.074  0.0218
##  1 - 3      -0.156 0.0803  6 -1.945  0.0997
##  2 - 3      -0.403 0.0803  6 -5.020  0.0024
```

```
# diagnostic plots
residplot(mm_3)
```

4.7 Linear regression between RY2 and CDS

```
# fit linear regression model
lm_3 <- lm(RY2 ~ CDS, data = a_new)
summary(lm_3)
```

```
##
## Call:
## lm(formula = RY2 ~ CDS, data = a_new)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.13338 -0.05085 -0.01090  0.06530  0.12439
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.9386111  0.0467991  20.056 2.09e-09 ***
## CDS         -0.0067778  0.0009845  -6.884 4.28e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.09061 on 10 degrees of freedom
## Multiple R-squared:  0.8258, Adjusted R-squared:  0.8083
## F-statistic: 47.39 on 1 and 10 DF,  p-value: 4.275e-05
```

```
anova(lm_3)
```

```
## Analysis of Variance Table
```

```
##
## Response: RY2
##           Df  Sum Sq Mean Sq F value    Pr(>F)
## CDS        1 0.38914 0.38914  47.394 4.275e-05 ***
## Residuals 10 0.08211 0.00821
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# diagnostic plots
residplot(lm_3)
```

Appendix A

SAS code

A.1 Exercise 4

Copy and paste the below code into a SAS editor, and hit run to see the output.

```
DATA A;
INPUT PLOT AN T BLK TRT PCTSEV;
Y=PCTSEV/100;
YSTAR=LOG(Y/(1-Y));
WT=Y*(1-Y);
DROP AN;
CARDS;
101 1   0   1   2   9
102 1   0   1   1   6
103 1   0   1   3   2
201 1   0   2   2   7
202 1   0   2   3   5
203 1   0   2   1   3
301 1   0   3   3   4
302 1   0   3   2   2
303 1   0   3   1   6
401 1   0   4   1   1
402 1   0   4   2   1
403 1   0   4   3   4
101 2   7   1   2   4
102 2   7   1   1   6
103 2   7   1   3   10
201 2   7   2   2   2
202 2   7   2   3   5
203 2   7   2   1   3
```

```
301 2   7   3   3   11
302 2   7   3   2   6
303 2   7   3   1   4
401 2   7   4   1   8
402 2   7   4   2   3
403 2   7   4   3   6
101 3   14  1   2   8
102 3   14  1   1   20
103 3   14  1   3   15
201 3   14  2   2   13
202 3   14  2   3   12
203 3   14  2   1   14
301 3   14  3   3   15
302 3   14  3   2   8
303 3   14  3   1   25
401 3   14  4   1   17
402 3   14  4   2   14
403 3   14  4   3   49
101 4   21  1   2   24
102 4   21  1   1   38
103 4   21  1   3   61
201 4   21  2   2   31
202 4   21  2   3   42
203 4   21  2   1   79
301 4   21  3   3   48
302 4   21  3   2   23
303 4   21  3   1   86
401 4   21  4   1   52
402 4   21  4   2   45
403 4   21  4   3   56
101 5   28  1   2   28
102 5   28  1   1   89
103 5   28  1   3   44
201 5   28  2   2   41
202 5   28  2   3   49
203 5   28  2   1   79
301 5   28  3   3   45
302 5   28  3   2   47
303 5   28  3   1   63
401 5   28  4   1   94
402 5   28  4   2   52
403 5   28  4   3   64
101 6   35  1   2   36
102 6   35  1   1   77
103 6   35  1   3   88
```

```
201 6   35  2   2   42
202 6   35  2   3   69
203 6   35  2   1   71
301 6   35  3   3   43
302 6   35  3   2   39
303 6   35  3   1   84
401 6   35  4   1   97
402 6   35  4   2   47
403 6   35  4   3   76
;
PROC MIXED DATA=A COVTEST;
CLASS BLK TRT;
MODEL YSTAR=TRT|T/ SOLUTION DDFM=bw RESIDUAL;
RANDOM BLK;
WEIGHT WT;
REPEATED/SUBJECT=BLK*TRT TYPE=AR(1) R RCORR;
quit;

PROC MIXED DATA=A;
CLASS BLK TRT;
MODEL YSTAR=TRT TRT*T/NOINT SOLUTION DDFM=bw OUTPM=B;
RANDOM BLK;
WEIGHT WT;
REPEATED/SUBJECT=BLK*TRT TYPE=AR(1);
LSMEANS TRT/DIFF AT T=0;
LSMEANS TRT/DIFF AT T=7;
LSMEANS TRT/DIFF AT T=14;
LSMEANS TRT/DIFF AT T=21;
LSMEANS TRT/DIFF AT T=28;
LSMEANS TRT/DIFF AT T=35;

ESTIMATE 'TRT1 S VS TRT2 S' TRT*T 1 -1 0;
ESTIMATE 'TRT1 S VS TRT3 S' TRT*T 1 0 -1;
ESTIMATE 'TRT2 S VS TRT3 S' TRT*T 0 1 -1;
quit;

PROC PRINT DATA=B;

PROC REG DATA=B;
MODEL YSTAR=PRED;

RUN;
```

A.2 Exercise 9.4

Copy and paste the below code into a SAS editor, and hit run to see the output.

```
DATA A;
INPUT I YI;
EAST=1;
NORTH=I;
CARDS;
1   41
2   60
3   81
4   22
5   8
6   20
7   28
8   2
9   0
10  2
11  2
12  8
13  0
14  43
15  61
16  50
;
PROC VARIOGRAM PLOTS=MORAN OUTVAR=B;
COMPUTE LAGD=1 MAXLAG=11 AUTOCORR(ASSUM=RANDOM);
COORDINATES XC=EAST YC=NORTH;
VAR YI;

PROC PRINT;
run;

PROC VARIOGRAM DATA=A PLOTS=FIT;
COMPUTE LAGD=1 MAXLAG=11 CL ROBUST;
COORDINATES XC=EAST YC=NORTH;
MODEL FORM=AUTO(MLIST=(SPH EXP GAU) NEST=1);
VAR YI;

RUN;
```

A.3 Exercise 9.5

Copy and paste the below code into a SAS editor, and hit run to see the output.

```
DATA A;
INPUT COL ROW YI;
CARDS;
1   1   2
2   1   2
3   1   0
4   1   3
5   1   1
6   1   1
7   1   1
8   1   5
9   1   22
10  1   13
11  1   14
12  1   6
1   2   2
2   2   0
3   2   0
4   2   3
5   2   0
6   2   2
7   2   7
8   2   54
9   2   57
10  2   49
11  2   42
12  2   2
1   3   3
2   3   1
3   3   0
4   3   1
5   3   0
6   3   9
7   3   6
8   3   62
9   3   94
10  3   75
11  3   7
12  3   2
1   4   33
2   4   3
```

```
3   4   0
4   4   2
5   4   0
6   4   20
7   4   25
8   4   79
9   4   95
10  4   32
11  4   12
12  4   2
1   5   4
2   5   1
3   5   3
4   5   2
5   5   6
6   5   23
7   5   14
8   5   64
9   5   31
10  5   9
11  5   13
12  5   16
1   6   0
2   6   2
3   6   1
4   6   4
5   6   4
6   6   5
7   6   4
8   6   9
9   6   10
10  6   19
11  6   6
12  6   1
1   7   0
2   7   7
3   7   7
4   7   9
5   7   4
6   7   12
7   7   7
8   7   7
9   7   12
10  7   13
11  7   11
```

```
12  7   2
1   8   0
2   8   2
3   8   11
4   8   19
5   8   12
6   8   32
7   8   11
8   8   9
9   8   31
10  8   67
11  8   27
12  8   30
1   9   5
2   9   10
3   9   35
4   9   56
5   9   62
6   9   45
7   9   21
8   9   18
9   9   43
10  9   94
11  9   77
12  9   33
1   10  11
2   10  24
3   10  78
4   10  100
5   10  99
6   10  68
7   10  52
8   10  45
9   10  74
10  10  98
11  10  99
12  10  37
1   11  7
2   11  29
3   11  79
4   11  97
5   11  92
6   11  95
7   11  100
8   11  89
```

```
9   11  53
10  11  46
11  11  50
12  11  16
1   12  7
2   12  22
3   12  31
4   12  50
5   12  56
6   12  79
7   12  100
8   12  61
9   12  53
10  12  36
11  12  33
12  12  2
;

PROC VARIOGRAM DATA=A PLOTS=MORAN OUTVAR=B;
COMPUTE LAGD=1 MAXLAG=12 AUTOCORR(ASSUM=RANDOM);
COORDINATES XC=COL YC=ROW;
VAR YI;

PROC PRINT;

PROC VARIOGRAM DATA=A PLOTS=FIT;
COMPUTE LAGD=1 MAXLAG=12 CL ROBUST;
COORDINATES XC=COL YC=ROW;
MODEL FORM=AUTO(MLIST=(SPH EXP GAU) NEST=1);
VAR YI;

RUN;
```

A.4 Yield loss

Copy and paste the below code into a SAS editor, and hit run to see the output.

```
DATA A;
INPUT WP SP $ BLK TRT FUNG DS YIELD;
CARDS;
101 A   1   1   0   43  205
101 B   1   1   1   1   399
102 A   1   2   1   2   426
```

```
102 B   1   2   0   92  102
103 A   1   3   1   2   385
103 B   1   3   0   7   355
201 A   2   2   1   4   412
201 B   2   2   0   75  224
202 A   2   3   1   3   425
202 B   2   3   0   10  352
203 A   2   1   0   27  330
203 B   2   1   1   2   375
301 A   3   1   1   2   420
301 B   3   1   0   47  303
302 A   3   3   0   12  342
302 B   3   3   1   1   395
303 A   3   2   0   63  222
303 B   3   2   1   3   393
401 A   4   3   0   12  326
401 B   4   3   1   1   400
402 A   4   1   0   32  289
402 B   4   1   1   1   418
403 A   4   2   1   2   371
403 B   4   2   0   53  175
;
PROC MIXED COVTEST METHOD=TYPE3;
CLASS BLK TRT FUNG;
MODEL DS=TRT|FUNG/RESIDUAL;
RANDOM BLK BLK*TRT;
LSMEANS TRT|FUNG/DIFF;
run;

PROC MIXED COVTEST METHOD=TYPE3;
CLASS BLK TRT FUNG;
MODEL YIELD=TRT|FUNG/RESIDUAL;
RANDOM BLK BLK*TRT;
LSMEANS TRT|FUNG/DIFF;

PROC REG;
MODEL YIELD=DS;

DATA B;
SET A;
RY1=YIELD/399.23843;

PROC REG;
MODEL RY1=DS;
```

```
PROC SORT DATA=A; BY BLK TRT FUNG;
PROC TRANSPOSE DATA=A OUT=T1A; BY BLK TRT;
VAR YIELD;
run;

DATA T2A;
SET T1A;
RY2=COL1/COL2;
DROP _NAME_ COL1 COL2;
run;

PROC TRANSPOSE DATA=A OUT=T1B; BY BLK TRT;
VAR DS;
run;

DATA T2B;
SET T1B;
CDS=COL1;
DROP _NAME_ COL1 COL2;
run;

DATA T3;
MERGE T2A T2B;
BY BLK TRT;

PROC PRINT;
run;

PROC MIXED COVTEST;
CLASS BLK TRT;
MODEL RY2=TRT/RESIDUAL;
RANDOM BLK;
LSMEANS TRT/DIFF;

PROC REG;
MODEL RY2=CDS;

RUN;
```