

# Linear modelling of soil temperature effects on root lesion nematode population densities in R

· 2 Comments ([https://openplantpathology.org/notes/sparks/1/#disqus\\_thread](https://openplantpathology.org/notes/sparks/1/#disqus_thread))

matode%20population%20densities%20in%20R&url=https%3a%2f%2fopenplantpathology.org%2fnotes%2fsparks%2f1%2f)

arks%2f1%2f)

%20modelling%20of%20soil%20temperature%20effects%20on%20root%20lesion%20nematode%20population%20densities%20in%20R)

g%20of%20soil%20temperature%20effects%20on%20root%20lesion%20nematode%20population%20densities%20in%20R)

0nematode%20population%20densities%20in%20R&body=https%3a%2f%2fopenplantpathology.org%2fnotes%2fsparks%2f1%2f)

## Adam H Sparks and John P Thompson

University of Southern Queensland, Centre for Crop Health, Toowoomba, Qld, AU

DOI 10.5281/zenodo.1173391 (<https://doi.org/10.5281/zenodo.1173391>) OPP Peer Reviewed

(<https://github.com/openplantpathology/contributions/issues/1>) release v1.0.4

([https://github.com/adamhsparks/Modelling\\_Nematode\\_Populations](https://github.com/adamhsparks/Modelling_Nematode_Populations))

## Introduction

*Pratylenchus thornei*, the root-lesion nematode, is widely distributed in wheat (*Triticum aestivum*) growing areas of many countries and is of particular concern in sub-tropical environments (Thompson 2015). These nematodes penetrate roots to feed and reproduce in the root cortex leading to loss of root function, which affects nutrient and water uptake of nutrients and water causing nutrient deficiency and water stress (Thompson 2015).

In the original paper the population response of *P. thornei* in Queensland, Australia wheat to temperature is modelled using a linear and quadratic equations. The study aimed to investigate the effects of soil profile temperatures after different sowing dates on reproduction of the nematodes in susceptible and moderately resistant wheat cultivars in the subtropical grain region of eastern Australia. This document recreates the models for population densities of *P. thornei* as described in the original paper.

## Objectives

There are two types of models described in the paper, the first model is a linear model used to describe the unplanted control and two quadratic models fit Gatcher (Susceptible) and GS50a (Moderately Resistant) wheat cultivars. For a more detailed discussion on fitting plant disease models in R, please see the “Linear Regression

(<http://www.apsnet.org/edcenter/advanced/topics/EcologyAndEpidemiologyInR/DiseaseProgress/Pages/LinearRegression.asp>)

module in the “Ecology and Epidemiology in R” documents available in the American Phytopathological Society's (APS) Education Center. For an even more in-depth discussion on linear models in R, how to fit and how to interpret the diagnostics that R provides the reader should refer to Faraway (2002).

This post will illustrate how to fit the original linear and quadratic models using the original data in R (R Core Team 2017).

## Packages

Using the **tidyverse** , (2017) package simplifies the libraries used in this work. It is a collection of packages designed to work together for data science, <https://www.tidyverse.org/> (<https://www.tidyverse.org/>). The **tidyverse** includes, **readr** (2017), used to import the data; **tidyr** (2018), used to format the data; **dplyr** (2017), used to subset the data; and **ggplot2** (2016), used for visualising the data and model fits. **viridis** (2018) is a selection of colour pallets that are widely accessible for people with colour-blindness and printing in black and white.

The following code chunk checks first to see if you have **tidyverse** , **viridis** and **hrbrthemes** installed, if not, it will automatically install them and then load them. Then set the default theme for all graphs to `theme_ipsum_rc` .

```
if (!require(tidyverse)) {
  install.packages(
    "tidyverse",
    repos = c(CRAN = "https://cloud.r-project.org/")
  )
  library(tidyverse)
}

if (!require(viridis)) {
  install.packages(
    "viridis",
    repos = c(CRAN = "https://cloud.r-project.org/")
  )
  library(viridis)
}

if (!require(hrbrthemes)) {
  install.packages(
    "hrbrthemes",
    repos = c(CRAN = "https://cloud.r-project.org/")
  )
  library(hrbrthemes)
}

ggplot2::theme_set(hrbrthemes::theme_ipsum_rc())
```

## Data Wrangling

The data are located in the `data` sub-folder. Import the data using `read_csv()` function from **readr** and view them.

```
nema <- read_csv("data/Nematode_Data.csv")

nema
```

Weeks<dbl>	Days<dbl>	Temperature<dbl>	Degree_days<dbl>	Unplanted<dbl>	Gatcher<dbl>	GS50a<dbl>	Potam<dbl>	Suneca<dbl>
8	56	15.0	280	5.748	6.773	6.691	7.613	6.703

8	56	20.0	560	5.915	9.513	7.420	9.285	9.121		
8	56	22.5	700	6.381	9.956	8.214	9.024	10.012		
8	56	25.0	840	6.510	9.354	8.254	9.732	9.202		
10	70	15.0	350	5.847	7.435	6.043	5.972	6.846		
10	70	20.0	700	6.157	10.338	8.915	10.284	10.175		
10	70	22.5	875	6.191	10.423	9.183	10.691	10.075		
10	70	25.0	1050	6.364	10.580	9.045	10.487	10.344		
12	84	15.0	420	5.755	9.926	8.187	8.745	9.573		
12	84	20.0	840	6.978	11.723	9.852	11.334	11.684		
1-10 of 24 rows						Previous	1	2	3	Next

```

nrow(nema)

## [1] 24

```

## Description of Fields in the Data

There are nine columns in the `nema` data described here in the following table.

Field	Data Description
Weeks	Number of weeks after wheat sowing
Days	Number of days after wheat sowing
Temperature	Temperature(°C) treatment
Degree_Days	Average thermal time degree days above 10 °C for four soil depths (8, 15, 30 and 60 cm)
Unplanted	Log*, log( ) , nematode population in the control treatment with no wheat planted
Gatcher	Log*, log( ) , nematode population in a susceptible wheat cultivar
GS50a	Log*, log( ) , nematode population in a moderately resistant wheat cultivar
Potam	Log*, log( ) , nematode population in a susceptible wheat cultivar
Suneca	Log*, log( ) , nematode population in a susceptible wheat cultivar

\* For an exploration into the reasons why the data were transformed using the natural log `log( )` , see the Exploring Why the Data Were Log Transformed in the Bonus Material section

## Wide to Long Data

You can see that each of the varieties have their own column in the original data format, this is commonly called wide data. Wide data are commonly found in spreadsheets but do not lend themselves easily to data analysis, modelling and visualisation. To make it easier to do these things it is common to convert the data from wide to long format, commonly referred to as tidying, when using R. The advantage of a tidy dataset is that it is easy to manipulate, model and visualize, and always has a specific structure where each variable is a column, each observation is a row, and each type of observational unit is a table (Wickham 2014).

In order to use **ggplot2** for visualising the data, they need to be converted from wide to long. Using `gather()` from the **tidyr** package to convert from wide to long format where the varieties are all listed in a single column, `Variety` .

```
nema_long <- nema %>% gather(Variety, Log_pop, Unplanted:Suneca)
```

```
nema_long
```

Weeks <dbl>	Days <dbl>	Temperature <dbl>	Degree_days <dbl>	Variety <chr>	Log_pop <dbl>
8	56	15.0	280	Unplanted	5.748
8	56	20.0	560	Unplanted	5.915
8	56	22.5	700	Unplanted	6.381
8	56	25.0	840	Unplanted	6.510
10	70	15.0	350	Unplanted	5.847
10	70	20.0	700	Unplanted	6.157
10	70	22.5	875	Unplanted	6.191
10	70	25.0	1050	Unplanted	6.364
12	84	15.0	420	Unplanted	5.755
12	84	20.0	840	Unplanted	6.978
1-10 of 120 rows				Previous	12

```
nrow(nema_long)
```

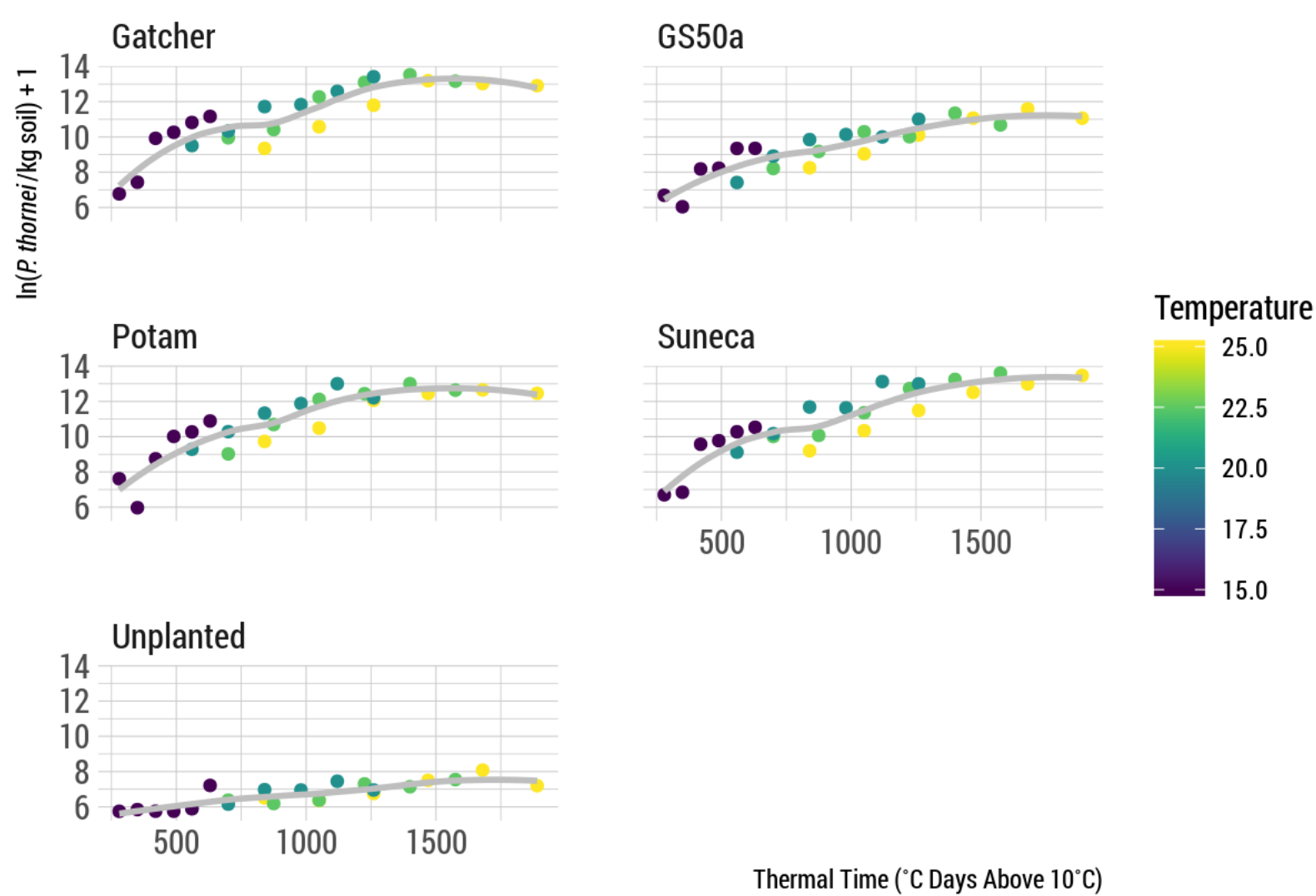
```
## [1] 120
```

As we see, the original `nema` data had only 24 rows and the long format of the data have 120 rows now.

## Data Visualisation

Now that the data are in the format that `ggplot2` uses, take a look at the data first to see what it looks like. Here we fit a smoothed line for each variety's nematode population to the raw data. The individual temperature treatments are shown here by shape, the variety by colour.

```
ggplot(
  nema_long,
  aes(
    x = Degree_days,
    y = Log_pop,
    colour = Temperature,
    group = Variety
  )
) +
  geom_point() +
  geom_smooth(
    colour = "grey",
    se = FALSE,
    alpha = 0.5
  ) +
  ylab(expression(
    paste(
      "ln(",
      italic("P. thornei"),
      "/kg soil) + 1"
    ),
    sep = ""
  )) +
  xlab("Thermal Time (°C Days Above 10°C)") +
  scale_colour_viridis("Temperature") +
  facet_wrap(~ Variety, ncol = 2)
```



# Modelling

## Unplanted Model

The paper uses a linear model for the unplanted control. Here we will write a function to use in modelling the unplanted population data. I have wrapped the model in a function which makes it pipe-able, `%>%` and has other advantages when it comes to fitting the same model to several sets of data.

In the linear equation for the Unplanted control treatment, the rate of population increase can be expressed as:

$$y = y_0 + rt$$

Where  $y_0$  is the initial population,  $r$  is the rate of change and  $t$  equal time.

## Fitting a Linear Model

```
linear_model <- function(df) {  
  lm(  
    Log_pop ~ Degree_days,  
    data = df  
  )  
}
```

Now check the model fit, using `filter()` from **dplyr** to select only Unplanted data from the data set for the model and fit the linear model to the data.

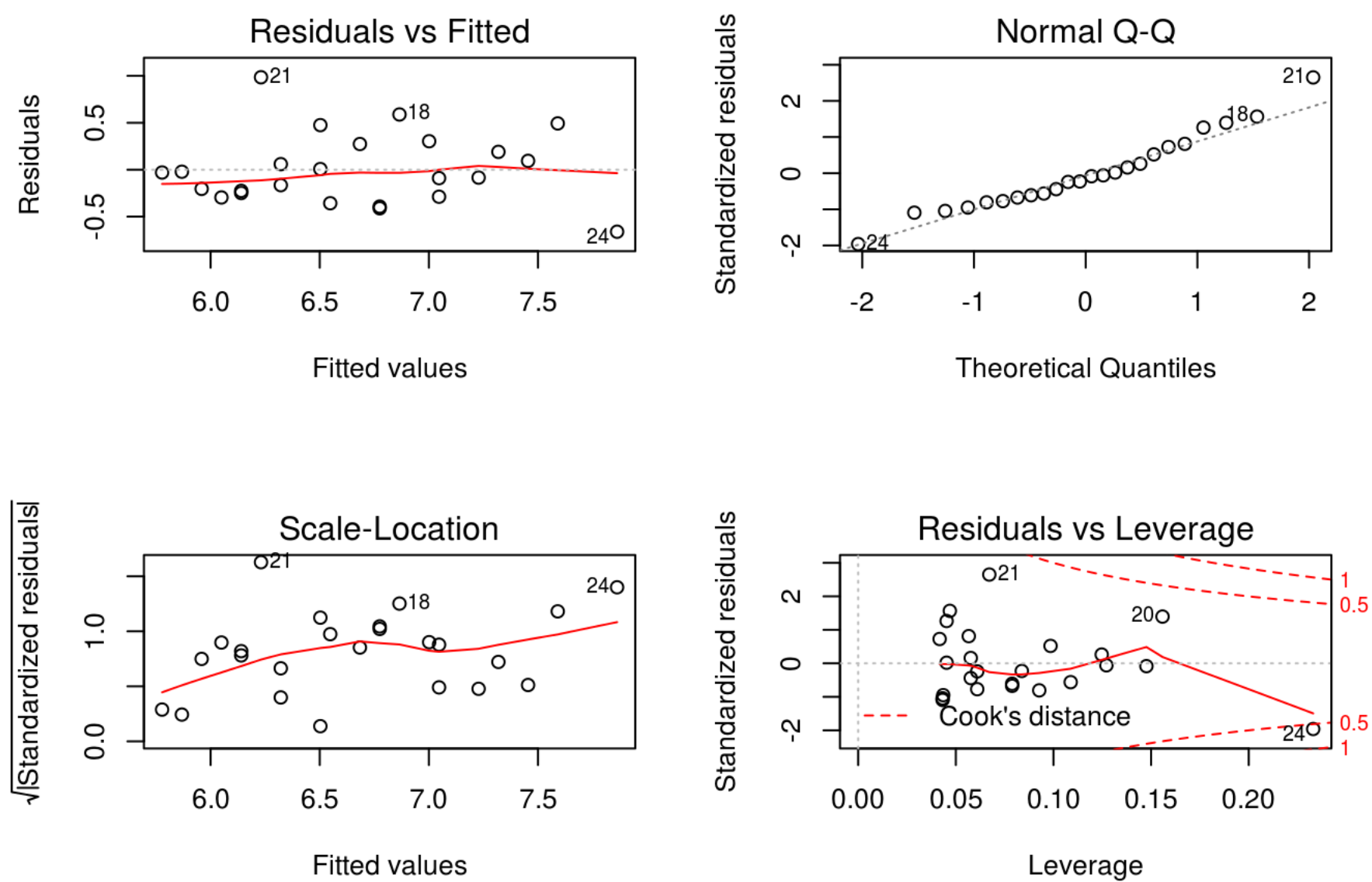
```
unplanted_model <- nema_long %>%  
  filter(Variety == "Unplanted") %>%  
  linear_model()
```

Using `par(mfrow = c(2, 2))` creates a four-panel graph rather than four individual graphs, which the next function will create by default.

Using the `plot()` function with any `lm()` object will create four diagnostic plots for your inspection.

```
par(mfrow = c(2, 2))  
plot(unplanted_model)
```





These plots do not appear to indicate anything amiss as one would hope for from the models that have already been published. If you are unfamiliar with how to interpret these diagnostic plots see [Interpreting Linear Models in R](#) in the Further Reading section.

Using the `summary()` function displays information about the model fit. If you are unfamiliar with how to read and interpret the output of `summary()` for a linear model, please refer to [Interpreting Linear Models in R](#) in the Further Reading section for references that go into more detail on this matter.

```
summary(unplanted_model)
```

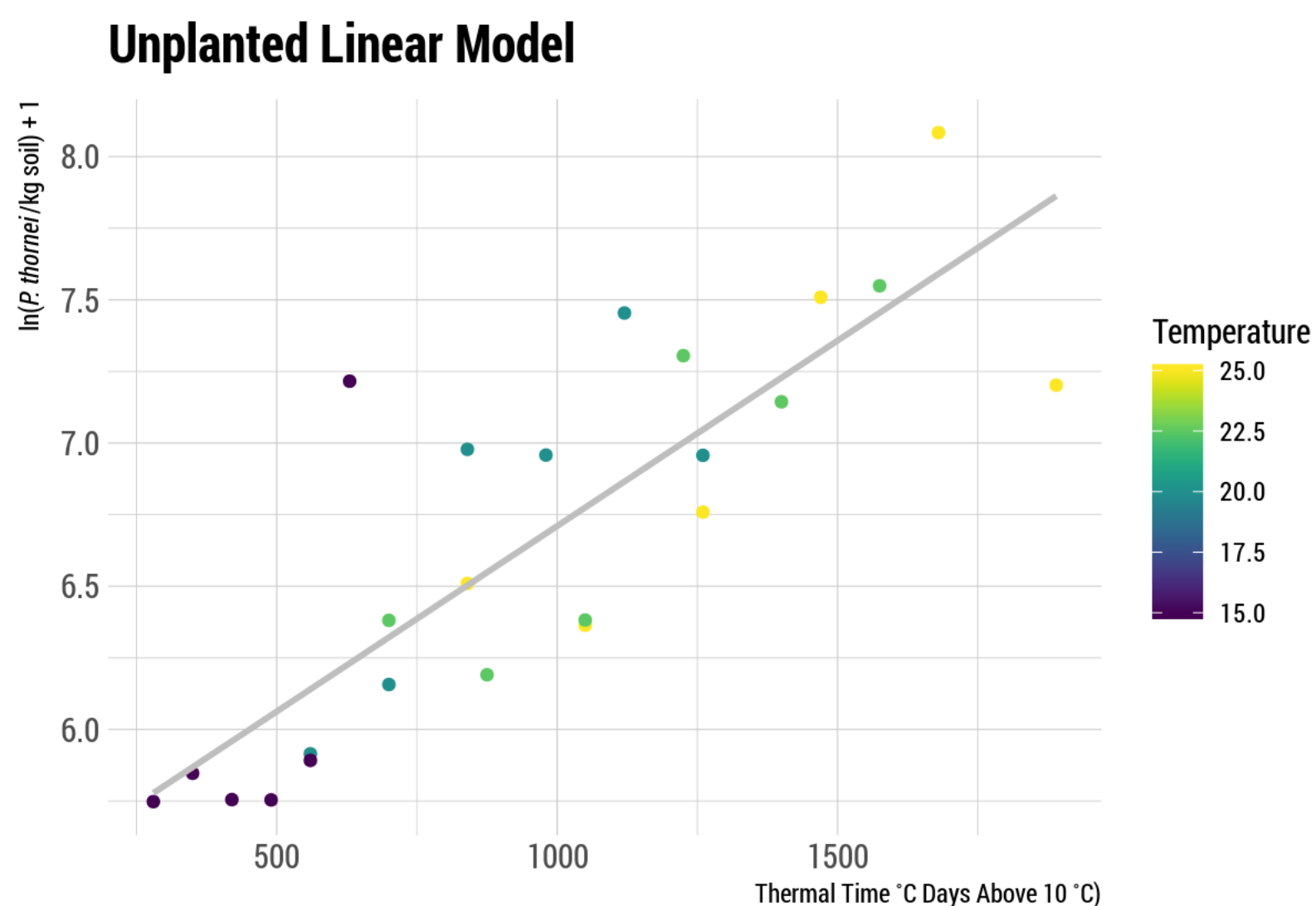
```
##
## Call:
## lm(formula = Log_pop ~ Degree_days, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.66053 -0.25811 -0.05683  0.21123  0.98511
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.4150643   0.1929731  28.061  < 2e-16 ***
## Degree_days  0.0012950   0.0001823   7.103 4.01e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3847 on 22 degrees of freedom
## Multiple R-squared:  0.6964, Adjusted R-squared:  0.6826
## F-statistic: 50.45 on 1 and 22 DF,  p-value: 4.006e-07
```

From the original paper, the  $R^2$  value of the unplanted linear model was 0.7, we can see here that agrees: 0.7. In the original paper,  $P < 0.001$ , R reports  $p - value : 4.006382710^{-7}$ , which also agrees.

## Visualising the Model Fit to the Data

Using **ggplot2**'s `geom_smooth()` we can fit the same model above and graph the resulting line.

```
nema_long %>%
  group_by(Variety) %>%
  filter(Variety == "Unplanted") %>%
  ggplot(aes(
    x = Degree_days,
    y = Log_pop,
    colour = Temperature
  )) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x,
    size = 1,
    se = FALSE,
    colour = "grey",
    alpha = 0.5
  ) +
  ylab(expression(
    paste(
      "ln(",
      italic("P. thornei"),
      "/kg soil) + 1"
    ),
    sep = ""
  )) +
  xlab("Thermal Time °C Days Above 10 °C") +
  scale_colour_viridis("Temperature") +
  ggtitle("Unplanted Linear Model")
```





# Quadratic Models

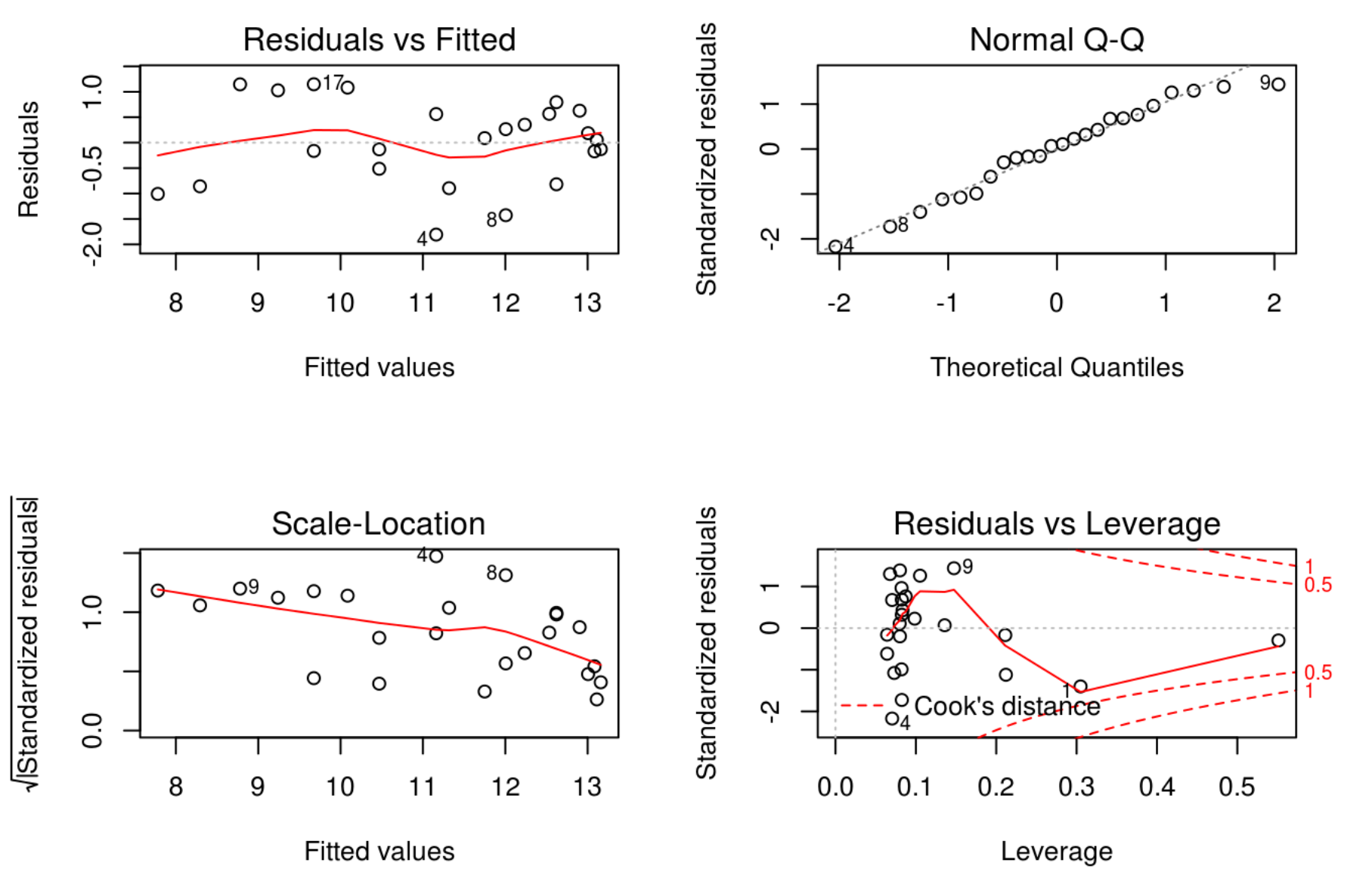
In the original paper, the quadratic model best described Gatcher and GS50a data, which are fit here.

```
quadratic_model <- function(df) {  
  lm(  
    Log_pop ~ Degree_days + I(Degree_days ^ 2),  
    data = df  
  )  
}
```

## Susceptible Varieties

Gatcher, Potam and Suneca all have very similar curves, here Gatcher is used to fit a quadratic model as in the original paper following the same methods as above for the linear model.

```
s_model <- nema_long %>%  
  filter(Variety == "Gatcher") %>%  
  quadratic_model()  
  
par(mfrow = c(2, 2))  
plot(s_model)
```



```
summary(s_model)
```

```
##
## Call:
## lm(formula = Log_pop ~ Degree_days + I(Degree_days^2), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.80668 -0.58936  0.07297  0.58228  1.14866
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.476e+00  9.043e-01   6.055 5.21e-06 ***
## Degree_days     8.961e-03  1.909e-03   4.693 0.000124 ***
## I(Degree_days^2) -2.612e-06  9.008e-07  -2.899 0.008579 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8631 on 21 degrees of freedom
## Multiple R-squared:  0.7998, Adjusted R-squared:  0.7808
## F-statistic: 41.96 on 2 and 21 DF,  p-value: 4.621e-08
```

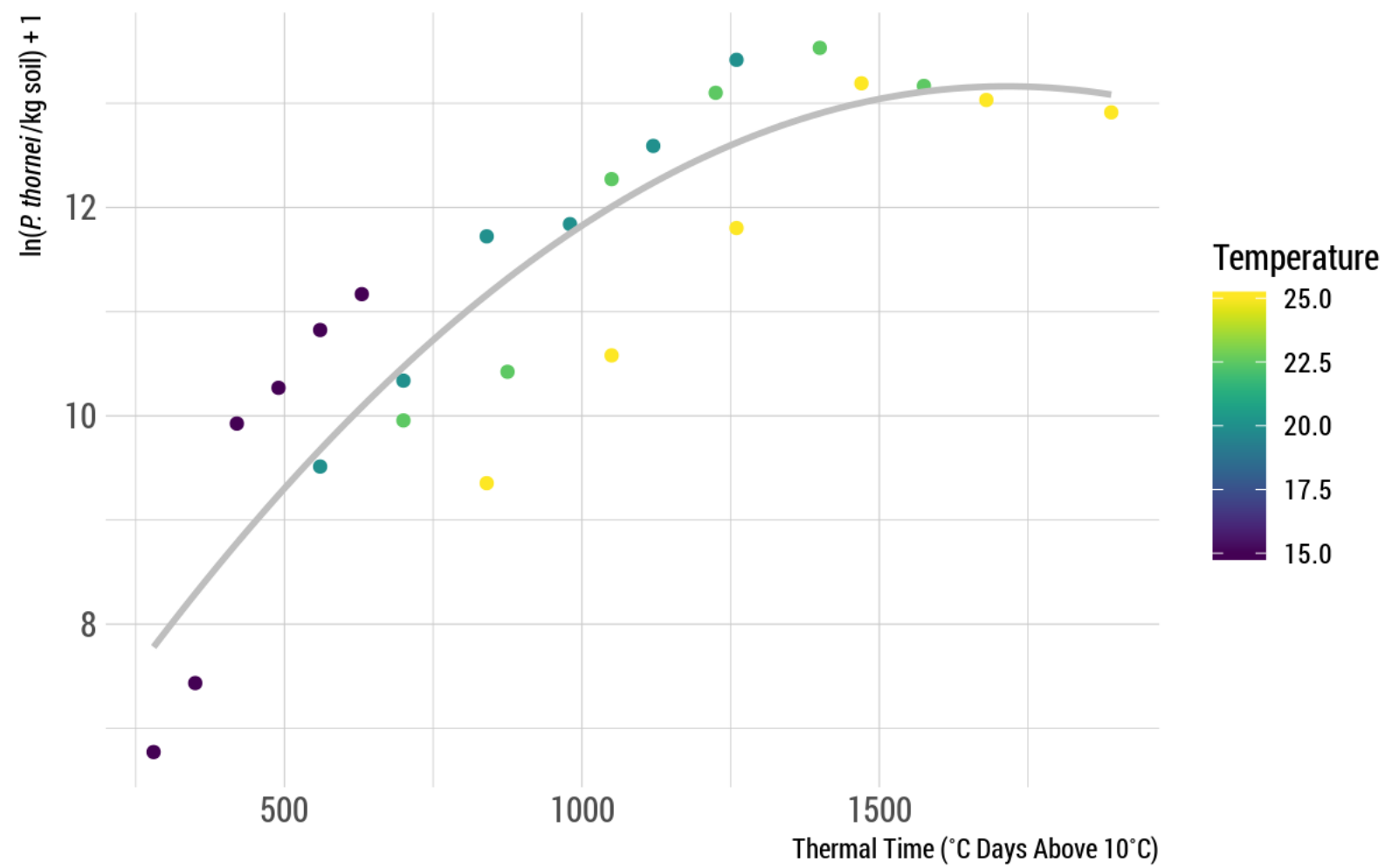
From the original paper, the  $R^2$  value of Gatcher's quadratic model was 0.80, we can see here that agrees: 0.8. In the original paper,  $P < 0.001$ , R reports  $p - value : 1.240363510^{-4}$ , which also agrees.

### Visualise Susceptible Variety Model

The model visualisation is the same for the quadratic models as the linear model, however you will note that the line has a downward curve at higher temperatures.

```
nema_long %>%
  group_by(Variety) %>%
  filter(Variety == "Gatcher") %>%
  ggplot(aes(
    x = Degree_days,
    y = Log_pop,
    colour = Temperature,
  )) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x + I(x ^ 2),
    size = 1,
    se = FALSE,
    colour = "grey",
    alpha = 0.5
  ) +
  ylab(expression(
    paste(
      "ln(",
      italic("P. thornei"),
      "/kg soil) + 1"
    ),
    sep = ""
  )) +
  xlab("Thermal Time (°C Days Above 10°C)") +
  scale_colour_viridis("Temperature") +
  ggtitle("Gatcher Quadratic Model")
```

## Gatcher Quadratic Model

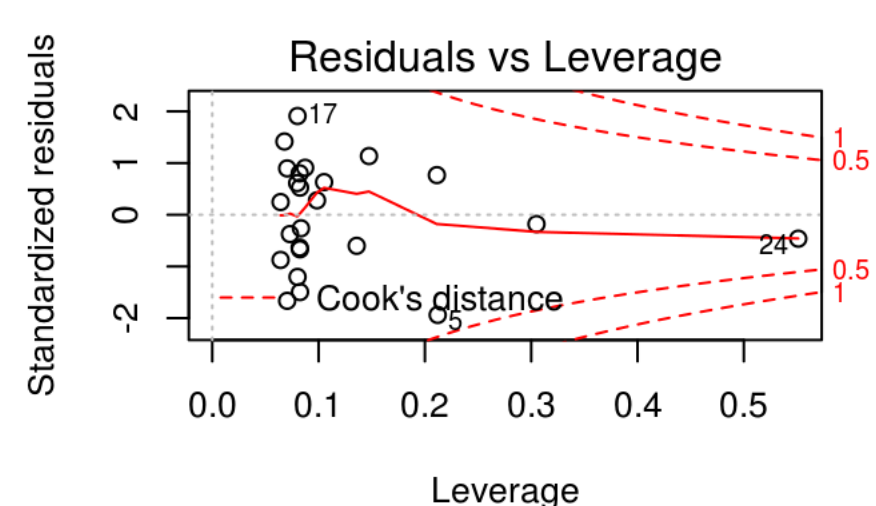
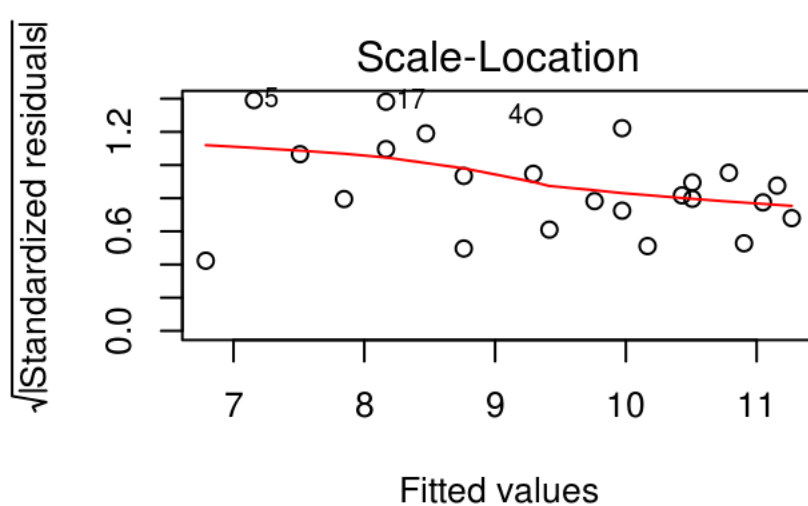
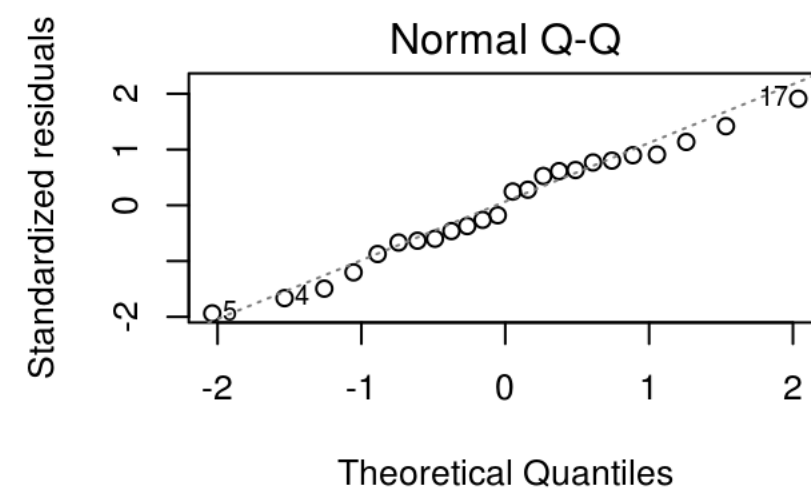
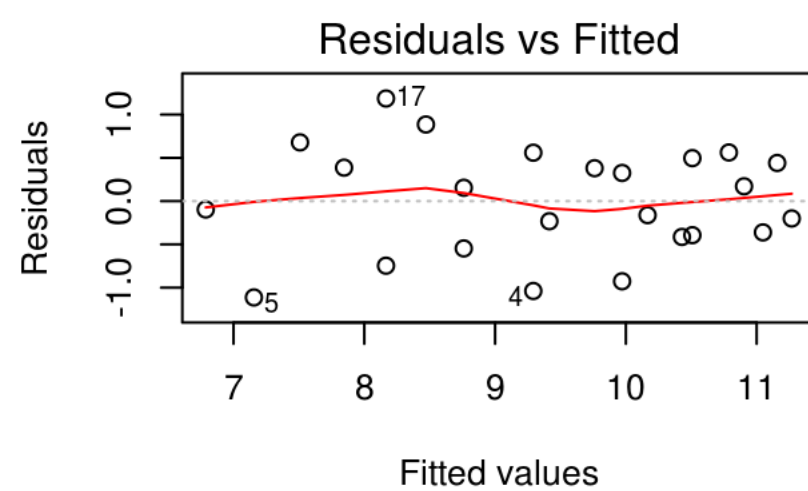


## Moderately Resistant Cultivar

GS50a, moderately resistant to *P. thornei*, also fits a quadratic model but the coefficients are slightly different due to different responses to the variety and temperature.

```
mr_model <- nema_long %>%
  filter(Variety == "GS50a") %>%
  quadratic_model()

par(mfrow = c(2, 2))
plot(mr_model)
```



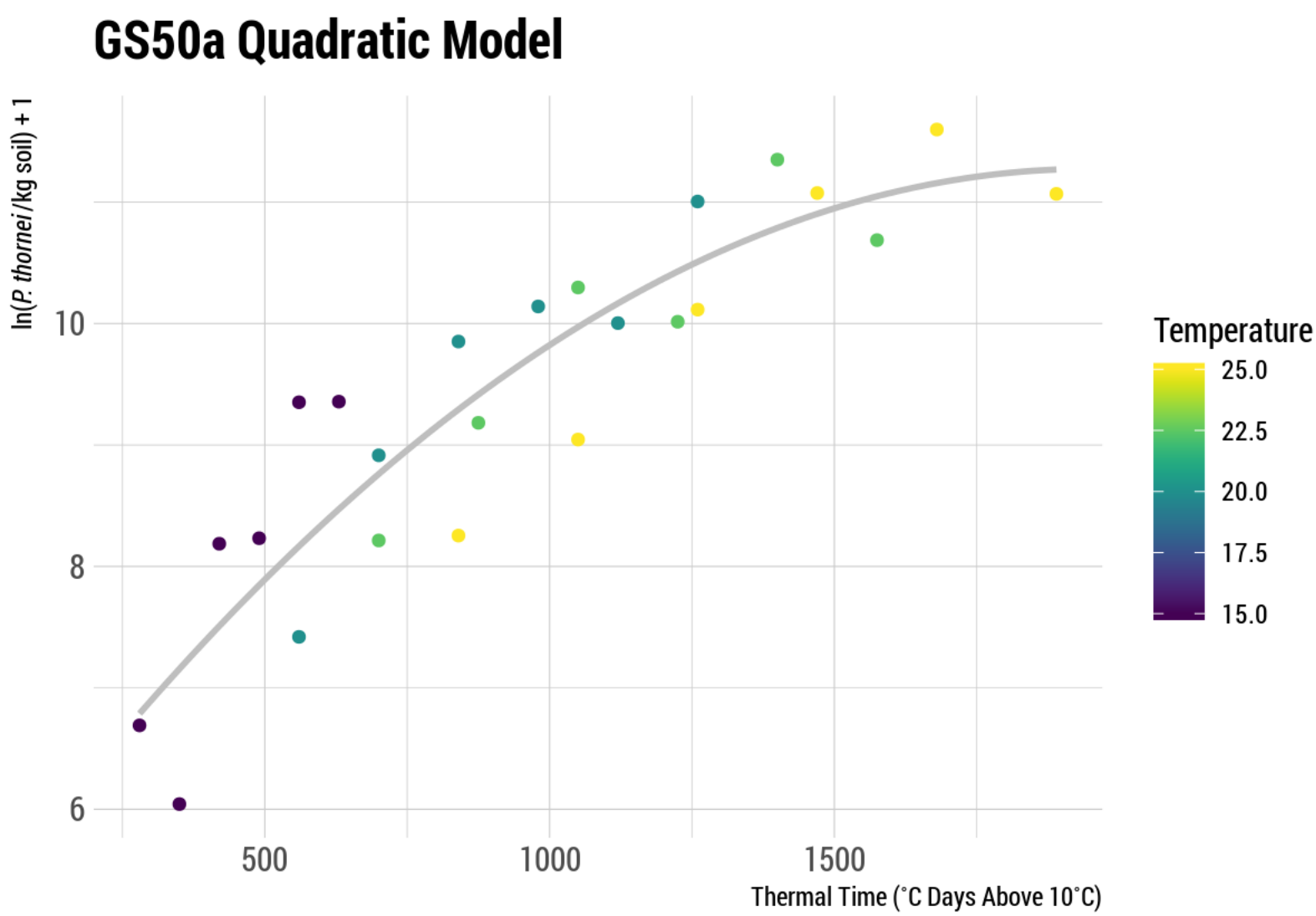
`summary(mr_model)`

```
##
## Call:
## lm(formula = Log_pop ~ Degree_days + I(Degree_days^2), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.11285 -0.39845  0.02889  0.45494  1.18598
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.157e+00  6.779e-01   7.607 1.83e-07 ***
## Degree_days     6.274e-03  1.431e-03   4.384  0.00026 ***
## I(Degree_days^2) -1.609e-06  6.753e-07  -2.383  0.02672 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.647 on 21 degrees of freedom
## Multiple R-squared:  0.8233, Adjusted R-squared:  0.8065
## F-statistic: 48.92 on 2 and 21 DF,  p-value: 1.248e-08
```

From the original paper, the  $R^2$  value of GS50a’s quadratic model was 0.82, we can see here that agrees: 0.82. In the original paper,  $P < 0.001$ , R reports  $p - value : 2.596147310^{-4}$ , which also agrees.

**Visualising the Model Fit to the Data**

```
nema_long %>%
  group_by(Variety) %>%
  filter(Variety == "GS50a") %>%
  ggplot(aes(
    x = Degree_days,
    y = Log_pop,
    colour = Temperature,
  )) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x + I(x ^ 2),
    size = 1,
    se = FALSE,
    colour = "grey",
    alpha = 0.5
  ) +
  ylab(expression(
    paste(
      "ln(",
      italic("P. thornei"),
      "/kg soil) + 1"
    ),
    sep = ""
  )) +
  xlab("Thermal Time (°C Days Above 10°C)") +
  scale_colour_viridis("Temperature") +
  ggtitle("GS50a Quadratic Model")
```



# Discussion and Conclusions

As in the original paper, the model equations can be derived from these models as well. The derived regression equations are:

Gatcher (Susceptible):

$$\ln(P.thornei + 1) = -0.000003(0.0000009)T^2 + 0.009(0.0019)T + 5.4671(0.904)$$

GS50a (Moderately Resistant):

$$\ln(P.thornei + 1) = -0.000002(0.0000007)T^2 + 0.0063(0.0014)T + 5.1559(0.678)$$

Unplanted Control:

$$\ln(P.thornei + 1) = 0.0013(0.00018)T + 5.4151(0.193)$$

Refer back to the `summary()` outputs for each of the models for the coefficient values and  $R^2$  values, which match those reported in the original paper where the models were fit with Genstat.

Gatcher and GS50a have similar phenologies, but differ in resistance to root lesion nematodes, making the model comparisons a reasonable objective. The original paper goes on to test the effect of sowing date based on degree days. (Thompson 2015) reported a 61% increase in yield on average from sowing the susceptible, intolerant cultivar Gatcher at the end of May than sowing it in the third week of June. By June the soil temperatures and nematode populations were both greater, leading to lower wheat yield. The effects were less pronounced in the moderately resistant cultivar, GS50a, but were similar with a reduction in nematode population densities occurring due to earlier planting.

The models illustrated here for Gatcher and GS50a were able to accurately reflect the changes in nematode population as a result of degree days, which affected the nematodes' ability to damage the crop and reduce yield (Thompson 2015).

# Bonus Material

## Exploring Why the Data Were Log Transformed

In the paper the the natural log,  $\ln() + 1$ , of the nematode population counts were used to fit the models. Here we will explore a bit further why this was necessary.

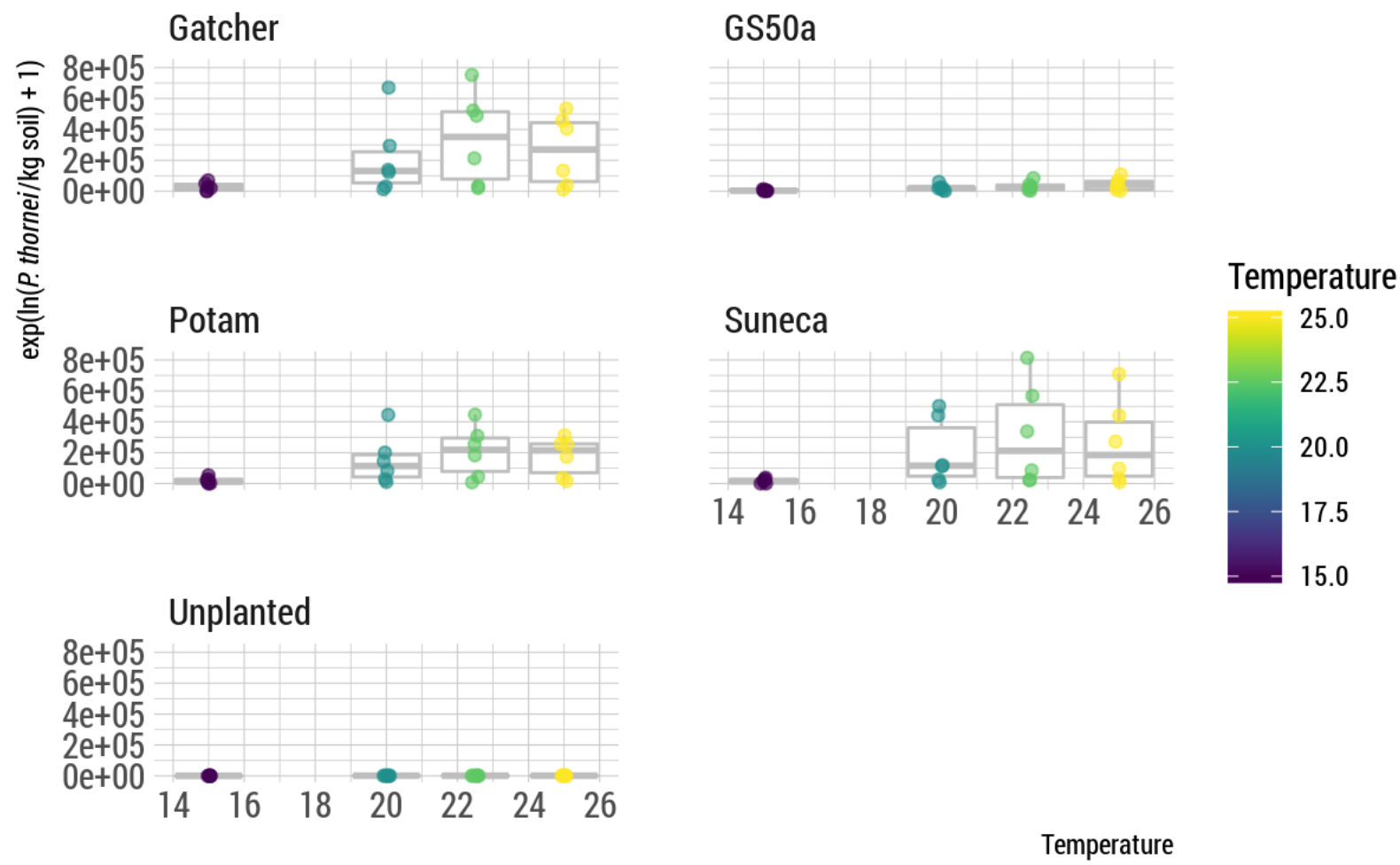
**i** A note about using  $\log() + 1$  rather than just  $\log()$ . This is necessary with these data to avoid taking  $\log(0)$ . Try it in R to see what happens if you are not familiar.

First, plot the data for each of the four temperatures and the four varieties, plus the unplanted control converting from the natural log value back to the original actual count values to see what the population numbers look like. Note the use of  $\exp() - 1$  in the `y aesthetic`, to transform the values from the  $\ln() + 1$  values. Doing this shows us the original data's values and helps demonstrate why the data were log transformed for analysis. To examine the data, first we will use boxplots and then quantile-quantile (qq) plots.

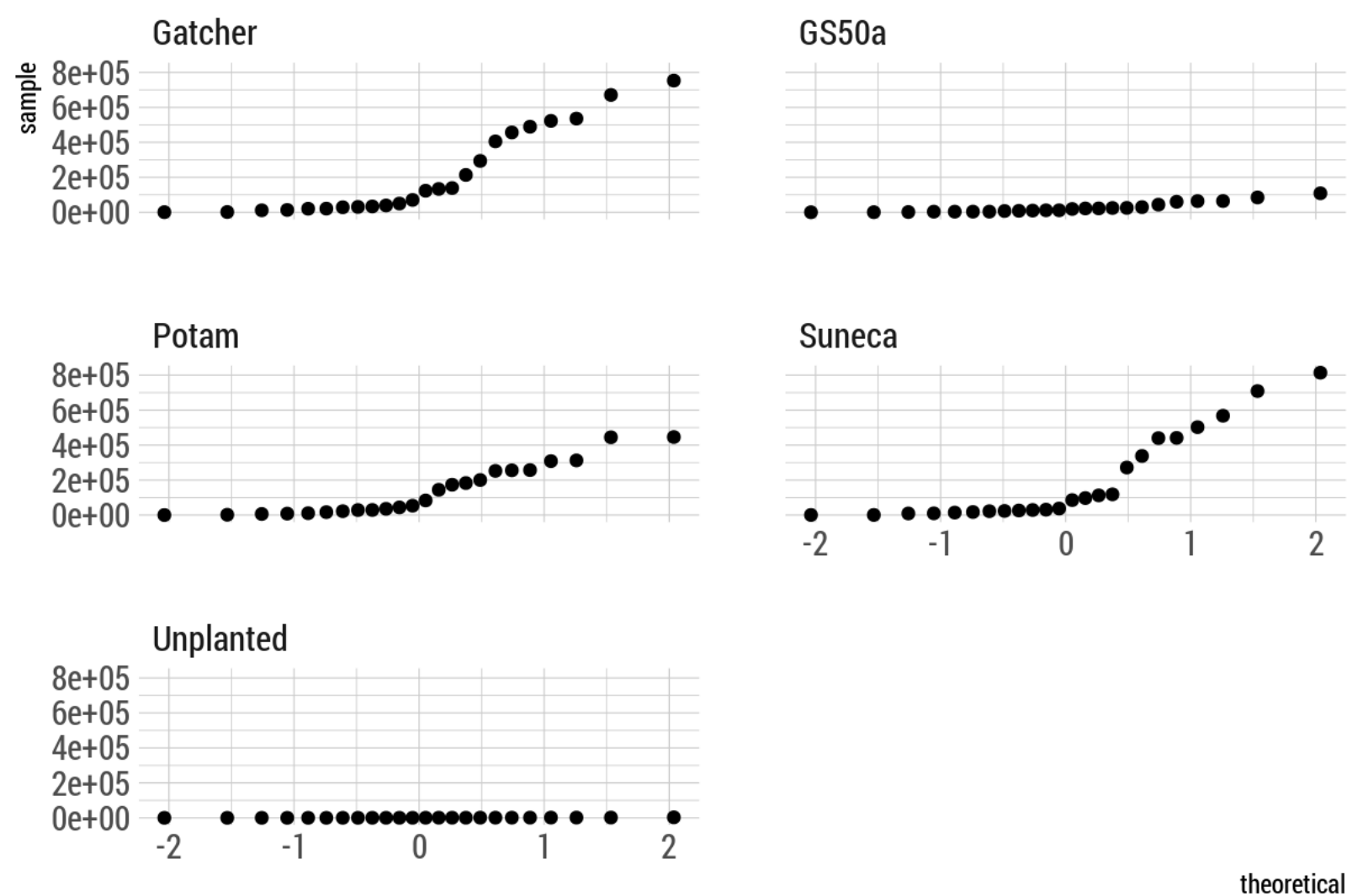


```
ggplot(
  nema_long,
  aes(
    x = Temperature,
    y = exp(Log_pop) - 1,
    group = Temperature,
    colour = Temperature
  )
) +
geom_boxplot(
  colour = "grey",
  outlier.color = NA
) +
geom_jitter(
  width = 0.1,
  alpha = 0.6
) +
ylab(expression(
  paste(
    "exp(ln(",
    italic("P. thornei"),
    "/kg soil) + 1)"
  ),
  sep = ""
)) +
facet_wrap(
  ~ Variety,
  ncol = 2
) +
scale_colour_viridis("Temperature") +
ggtitle("Untransformed Data")
```

Untransformed Data



```
ggplot(
  nema_long,
  aes(sample = exp(Log_pop) - 1)
) +
  stat_qq() +
  facet_wrap(
    ~ Variety,
    ncol = 2
  )
```

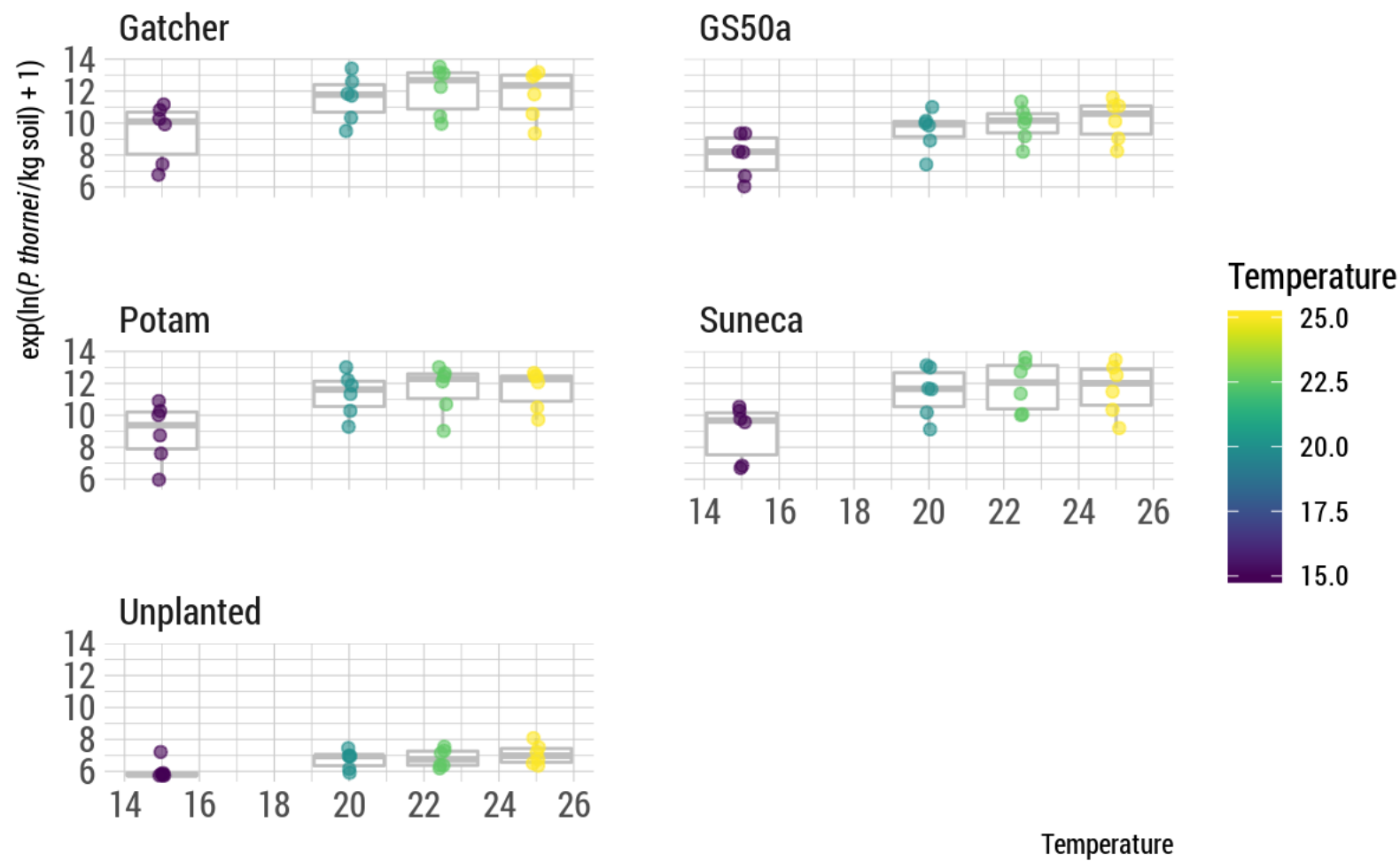


The boxplots show that there is a wide range of values with the 25 °C temperature populations close to zero with others having quite large ranges, this could indicate heteroscedasticity.

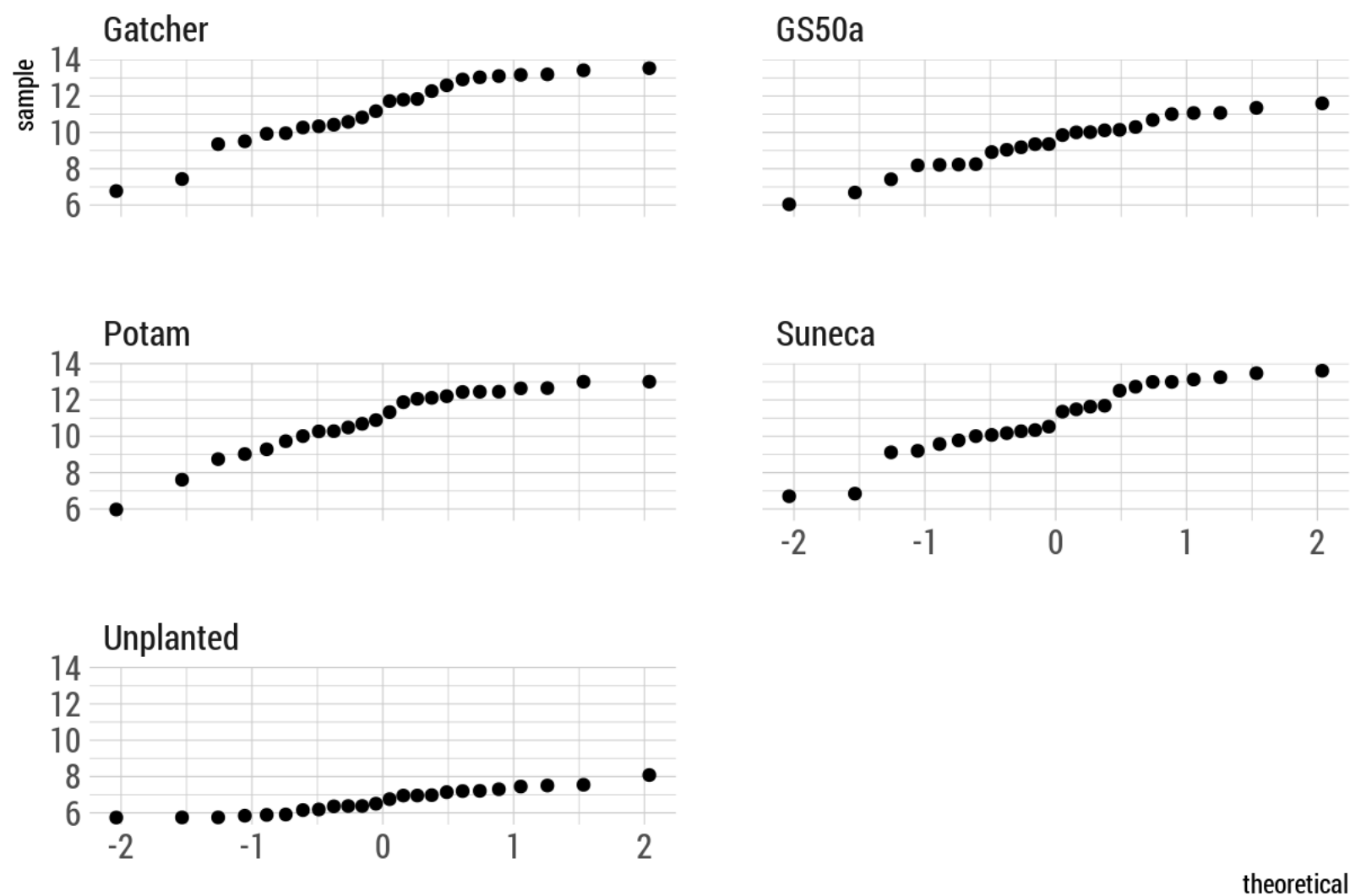
Also, looking at the qq-plots it is apparent that the original data do not meet the assumptions of normally distributed errors for a linear model. See the Further Reading section for suggested reading on interpreting qq-plots.

```
ggplot(
  nema_long,
  aes(
    x = Temperature,
    y = Log_pop,
    group = Temperature,
    colour = Temperature
  )
) +
  geom_boxplot(
    colour = "grey",
    outlier.color = NA
  ) +
  geom_jitter(
    width = 0.1,
    alpha = 0.6
  ) +
  ylab(expression(
    paste(
      "exp(ln(",
      italic("P. thornei"),
      "/kg soil) + 1)"
    ),
    sep = ""
  )) +
  facet_wrap(
    ~ Variety,
    ncol = 2
  ) +
  scale_colour_viridis("Temperature") +
  ggtitle("Log Transformed Data")
```

Log Transformed Data



```
ggplot(
  nema_long,
  aes(sample = Log_pop)
) +
  stat_qq() +
  facet_wrap(
    ~ Variety,
    ncol = 2
  )
```



Here we see that the `log()` transformed data's boxplots show fewer outliers and tighter range of values. The qq-plots also indicate that it is possible to conduct a linear regression with these data.

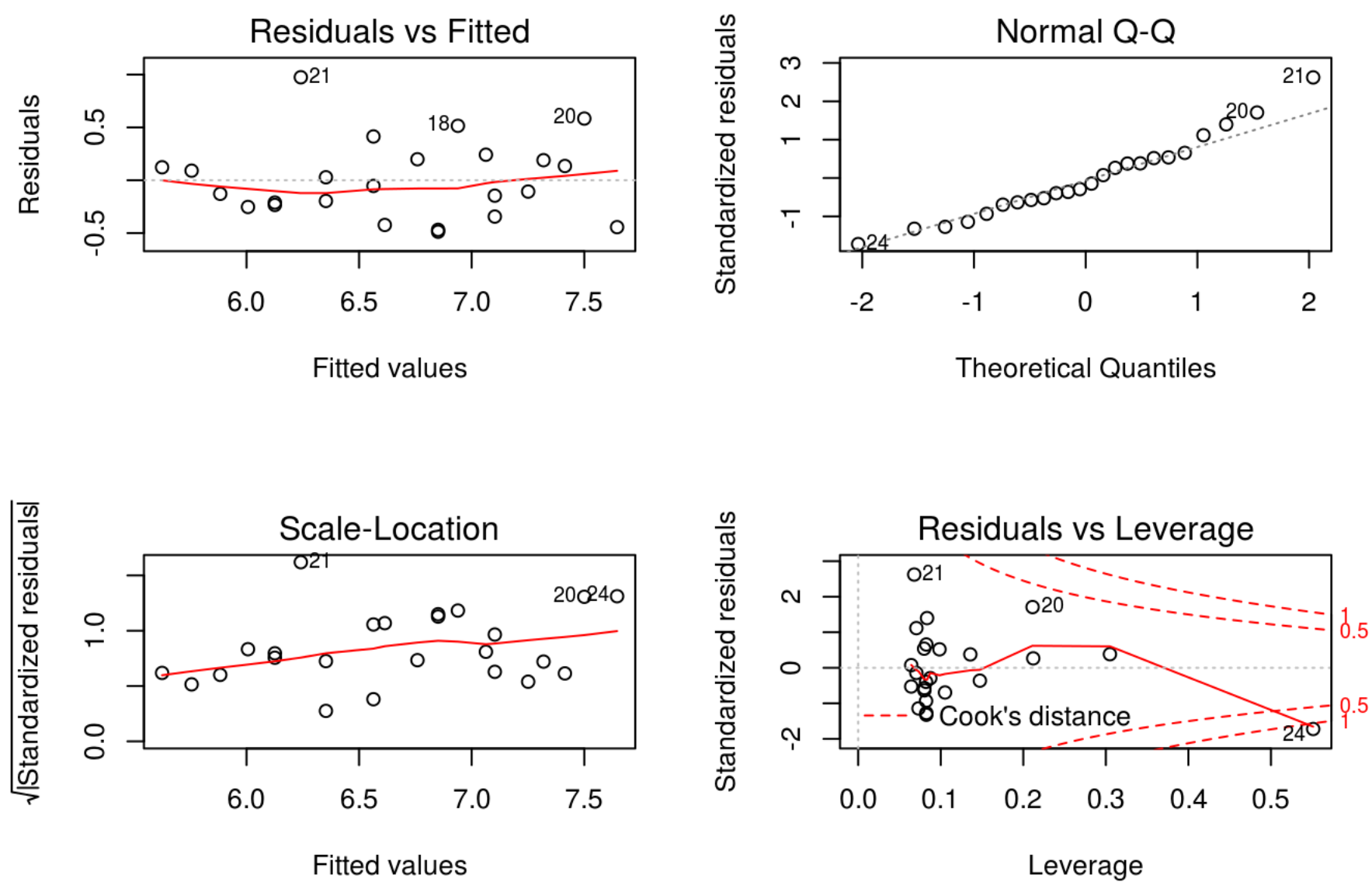
## Using AIC to Compare Model Quality

Even though the original paper used a linear model for the unplanted data, a polynomial model also fits these data quite well. We can compare the original linear model from the paper with a polynomial model quite easily in R to see how the models compare using AIC (Akaike information criterion). AIC is used to measure the models' relative quality to each other.

Since the `unplanted_model` object already exists as a product of the linear model, we simply need to use the polynomial model with the unplanted data to create a new object to compare them.

```
unplanted_poly_model <- nema_long %>%
  filter(Variety == "Unplanted") %>%
  quadratic_model()

par(mfrow = c(2, 2))
plot(unplanted_poly_model)
```

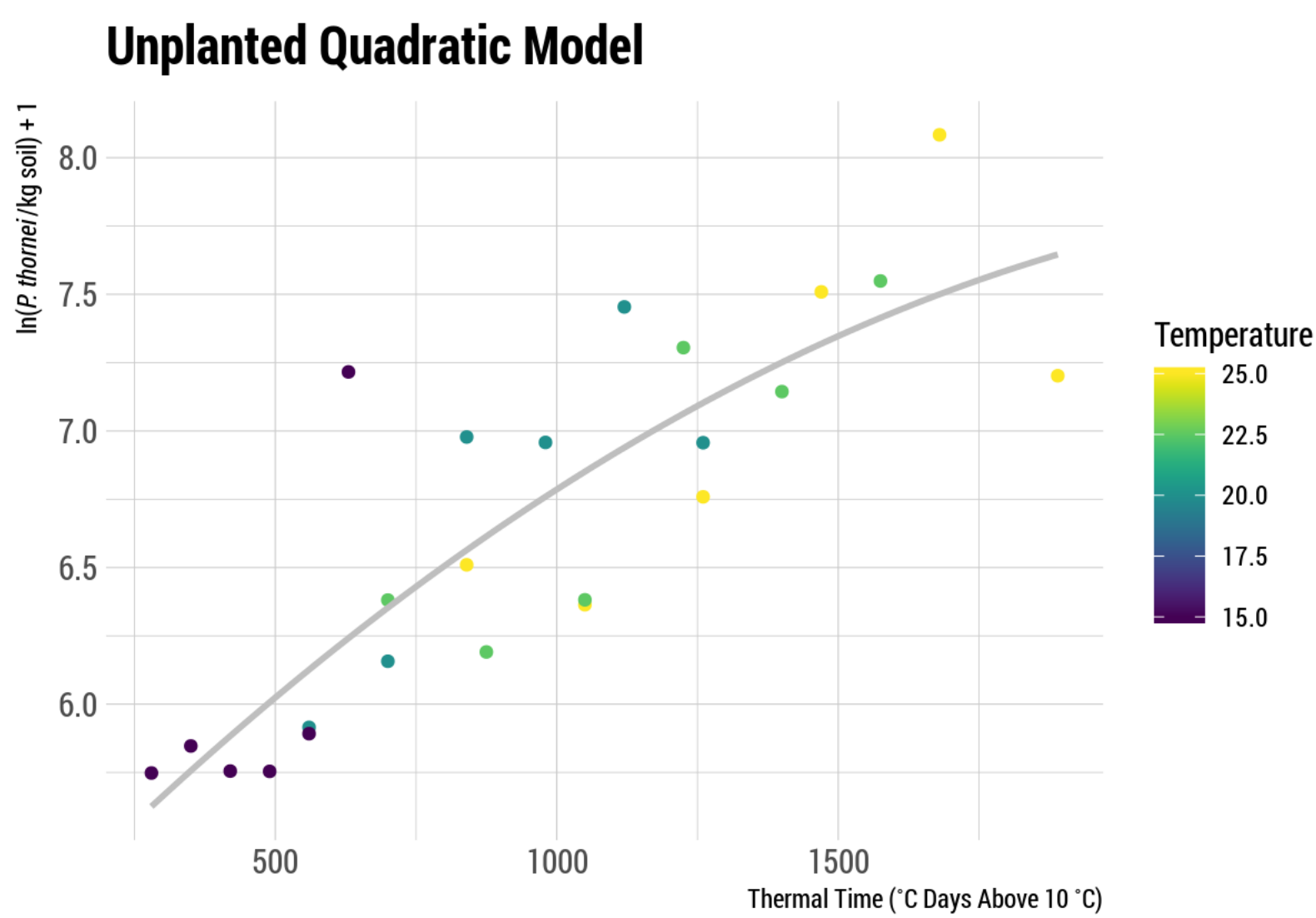


```
summary(unplanted_poly_model)
```

```
##
## Call:
## lm(formula = Log_pop ~ Degree_days + I(Degree_days^2), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48697 -0.23865 -0.08038  0.19211  0.97466
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)    5.062e+00  4.031e-01  12.556 3.14e-11 ***
## Degree_days     2.125e-03  8.511e-04   2.497  0.0209 *
## I(Degree_days^2) -4.010e-07  4.016e-07  -0.999  0.3293
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3848 on 21 degrees of freedom
## Multiple R-squared:  0.7101, Adjusted R-squared:  0.6825
## F-statistic: 25.72 on 2 and 21 DF,  p-value: 2.255e-06
```

By this information, the  $R^2$  value is a bit better from the `unplanted_poly_model`, 0.7101253, than the original `unplanted_model`'s, 0.6963592. Using the same code from above it is easy to visualise the new model's fit using **ggplot2**.

```
nema_long %>%
  group_by(Variety) %>%
  filter(Variety == "Unplanted") %>%
  ggplot(aes(
    x = Degree_days,
    y = Log_pop,
    colour = Temperature,
  )) +
  geom_point() +
  geom_smooth(
    method = "lm",
    formula = y ~ x + I(x ^ 2),
    size = 1,
    se = FALSE,
    colour = "grey",
    alpha = 0.5
  ) +
  ylab(expression(
    paste(
      "ln(",
      italic("P. thornei"),
      "/kg soil) + 1"
    ),
    sep = ""
  )) +
  xlab("Thermal Time (°C Days Above 10 °C)") +
  scale_colour_viridis("Temperature") +
  ggtitle("Unplanted Quadratic Model")
```



Checking the model fit visually, we can see that it fits the data nicely. To get a better feel for how these models compare, AIC can be used to determine the relative quality of a model for a *given set of data*. That is, you cannot compare models for other data using AIC.



Checking the AIC is quite simple in R, just `AIC()` . Here we check the AIC of the original linear `unplanted_model` and the new `unplanted_poly_model` .

```
AIC(unplanted_model)

## [1] 26.17149
```

```
AIC(unplanted_poly_model)

## [1] 27.05797
```

Ideally when fitting models, you look for the least complex model that provides the best explanation of the variation in the data. In this case the original linear model has a lower AIC, 26.1714857, than that of the polynomial model, 27.0579669, but they are extremely close and the  $R^2$  value of the polynomial model, 0.7101253, is a bit better than the linear model's  $R^2$ , 0.6963592, as well. Therefore, without more data to distinguish the models it appears that either model suffices for the data provided.

## Further Reading

### Tidy Data

Wickham (2014) introduced the idea of tidy data for analysis. As you work with raw data from many sources, it is useful to understand what this means and why it is useful. In this example, **tidyr** was used to convert the data from wide to long format. For a more in-depth look at using **tidyr** see:

- [Introducing tidyr \(https://blog.rstudio.com/2014/07/22/introducing-tidyr/\)](https://blog.rstudio.com/2014/07/22/introducing-tidyr/)
- [Gather columns into key-value pairs \(http://tidyr.tidyverse.org/reference/gather.html\)](http://tidyr.tidyverse.org/reference/gather.html).

### Interpreting Linear Models in R

The University of Georgia has a nice, easy to understand set of materials that demonstrate how to interpret diagnostic plot outputs from `plot(lm.object)` , Regression diagnostic plots (<http://strata.uga.edu/8370/rtips/regressionPlots.html>) on their Data Analysis in the Geosciences page. For even more, this Cross Validated question has an excellent discussion on Interpreting `plot.lm()` (<https://stats.stackexchange.com/questions/58141/interpreting-plot-lm>).

The University of Montana provides an on-line text, “*Statistics With R*”, that includes a section on ANOVA model diagnostics including QQ-plots (<https://arc.lib.montana.edu/book/statistics-with-r-textbook/item/57>). Since ANOVA uses `lm()` in R, the tools and descriptions here are applicable to the qq-plots we have generated here in this illustration.

For a detailed look at how to interpret the output from `summary()` for linear models, see The YHAT Blog post, Fitting & Interpreting Linear Models in R (<http://blog.yhat.com/posts/r-lm-summary.html>).

Faraway (2002), “*Practical Regression and Anova using R* (<https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf>)” is an excellent free resource that goes into detail about fitting linear models using R and how to interpret the diagnostics. Prof. Faraway has more recent books on the subject as well that you might wish to borrow from your library or purchase, see <http://www.maths.bath.ac.uk/~jjf23/LMR/> (<http://www.maths.bath.ac.uk/~jjf23/LMR/>) for more details.

## Selecting the Right Colour Scheme

Selecting good colour schemes is essential for communicating your message. The **viridis** package makes this much easier to do. Bob Rudis has a nice blog post when the package was first introduced that demonstrates why it is useful to use a package like this for your colour palettes, Using the new ‘viridis’ colormap in R (thanks to Simon Garnier) (<https://rud.is/b/2015/07/20/using-the-new-viridis-colormap-in-r-thanks-to-simon-garnier/>). Other colour palettes for R exist as well. Notably the **RColorBrewer** package provides an easy-to-use interface for the fantastic Colour Brewer palettes <http://colorbrewer2.org/> (<http://colorbrewer2.org/>) commonly used for cartography but also useful for graphs.

## Reproducibility

```
## — Session info —
##   setting  value
##   version  R version 3.5.2 (2017-01-27)
##   os       Ubuntu 14.04.5 LTS
##   system   x86_64, linux-gnu
##   ui       X11
##   language (EN)
##   collate  en_US.UTF-8
##   ctype    en_US.UTF-8
##   tz       UTC
##   date     2019-01-13
##
## — Packages —
##   package      * version      date      lib source
##   assertthat    0.2.0        2017-04-11 [1] CRAN (R 3.5.2)
##   backports     1.1.3        2018-12-14 [1] CRAN (R 3.5.2)
##   bindr         0.1.1        2018-03-13 [1] CRAN (R 3.5.2)
##   bindrcpp      * 0.2.2        2018-03-29 [1] CRAN (R 3.5.2)
##   blogdown      0.10         2019-01-09 [1] CRAN (R 3.5.2)
##   bookdown      0.9          2018-12-21 [1] CRAN (R 3.5.2)
##   broom         0.5.1        2018-12-05 [1] CRAN (R 3.5.2)
##   callr         3.1.1        2018-12-21 [1] CRAN (R 3.5.2)
##   cellranger    1.1.0        2016-07-27 [1] CRAN (R 3.5.2)
##   cli           1.0.1        2018-09-25 [1] CRAN (R 3.5.2)
##   colorspace    1.3-2        2016-12-14 [1] CRAN (R 3.5.2)
##   crayon        1.3.4        2017-09-16 [1] CRAN (R 3.5.2)
##   desc          1.2.0        2018-05-01 [1] CRAN (R 3.5.2)
##   devtools      2.0.1        2018-10-26 [1] CRAN (R 3.5.2)
##   digest        0.6.18       2018-10-10 [1] CRAN (R 3.5.2)
##   dplyr         * 0.7.8        2018-11-10 [1] CRAN (R 3.5.2)
##   evaluate      0.12         2018-10-09 [1] CRAN (R 3.5.2)
##   extrafont     0.17         2014-12-08 [1] CRAN (R 3.5.2)
##   extrafontdb   1.0          2012-06-11 [1] CRAN (R 3.5.2)
##   forcats       * 0.3.0        2018-02-19 [1] CRAN (R 3.5.2)
##   fs            1.2.6        2018-08-23 [1] CRAN (R 3.5.2)
##   generics      0.0.2        2018-11-29 [1] CRAN (R 3.5.2)
##   ggplot2       * 3.1.0        2018-10-25 [1] CRAN (R 3.5.2)
##   glue          1.3.0        2018-07-17 [1] CRAN (R 3.5.2)
##   gridExtra     2.3          2017-09-09 [1] CRAN (R 3.5.2)
##   gtable        0.2.0        2016-02-26 [1] CRAN (R 3.5.2)
##   haven         2.0.0        2018-11-22 [1] CRAN (R 3.5.2)
##   hms           0.4.2        2018-03-10 [1] CRAN (R 3.5.2)
##   hrbrthemes    * 0.5.0.1     2018-08-19 [1] CRAN (R 3.5.2)
```

```
## htmltools      0.3.6      2017-04-28 [1] CRAN (R 3.5.2)
## httr           1.4.0      2018-12-11 [1] CRAN (R 3.5.2)
## jsonlite       1.6        2018-12-07 [1] CRAN (R 3.5.2)
## kableExtra     * 0.9.0      2018-05-21 [1] CRAN (R 3.5.2)
## klippy         0.0.0.9500 2019-01-09 [1] Github (RLesur/klippy@201d363)
## knitr          1.21       2018-12-10 [1] CRAN (R 3.5.2)
## labeling       0.3        2014-08-23 [1] CRAN (R 3.5.2)
## lattice        0.20-38    2018-11-04 [3] CRAN (R 3.5.2)
## lazyeval       0.2.1      2017-10-29 [1] CRAN (R 3.5.2)
## lubridate      1.7.4      2018-04-11 [1] CRAN (R 3.5.2)
## magrittr       1.5        2014-11-22 [1] CRAN (R 3.5.2)
## memoise        1.1.0      2017-04-21 [1] CRAN (R 3.5.2)
## modelr         0.1.2      2018-05-11 [1] CRAN (R 3.5.2)
## munsell        0.5.0      2018-06-12 [1] CRAN (R 3.5.2)
## nlme           3.1-137    2018-04-07 [3] CRAN (R 3.5.2)
## pillar         1.3.1      2018-12-15 [1] CRAN (R 3.5.2)
## pkgbuild       1.0.2      2018-10-16 [1] CRAN (R 3.5.2)
## pkgconfig      2.0.2      2018-08-16 [1] CRAN (R 3.5.2)
## pkgload        1.0.2      2018-10-29 [1] CRAN (R 3.5.2)
## plyr           1.8.4      2016-06-08 [1] CRAN (R 3.5.2)
## prettyunits    1.0.2      2015-07-13 [1] CRAN (R 3.5.2)
## processx       3.2.1      2018-12-05 [1] CRAN (R 3.5.2)
## ps             1.3.0      2018-12-21 [1] CRAN (R 3.5.2)
## purrr          * 0.2.5      2018-05-29 [1] CRAN (R 3.5.2)
## R6             2.3.0      2018-10-04 [1] CRAN (R 3.5.2)
## Rcpp           1.0.0      2018-11-07 [1] CRAN (R 3.5.2)
## readr          * 1.3.1      2018-12-21 [1] CRAN (R 3.5.2)
## readxl         1.2.0      2018-12-19 [1] CRAN (R 3.5.2)
## remotes        2.0.2      2018-10-30 [1] CRAN (R 3.5.2)
## rlang          0.3.1      2019-01-08 [1] CRAN (R 3.5.2)
## rmarkdown      1.11       2018-12-08 [1] CRAN (R 3.5.2)
## rprojroot      1.3-2      2018-01-03 [1] CRAN (R 3.5.2)
## rstudioapi     0.9.0      2019-01-09 [1] CRAN (R 3.5.2)
## Rttf2pt1       1.3.7      2018-06-29 [1] CRAN (R 3.5.2)
## rvest          0.3.2      2016-06-17 [1] CRAN (R 3.5.2)
## scales         1.0.0      2018-08-09 [1] CRAN (R 3.5.2)
## sessioninfo    1.1.1      2018-11-05 [1] CRAN (R 3.5.2)
## stringi        1.2.4      2018-07-20 [1] CRAN (R 3.5.2)
## stringr        * 1.3.1      2018-05-10 [1] CRAN (R 3.5.2)
## tibble         * 2.0.1      2019-01-12 [1] CRAN (R 3.5.2)
## tidyr          * 0.8.2      2018-10-28 [1] CRAN (R 3.5.2)
## tidyselect     0.2.5      2018-10-11 [1] CRAN (R 3.5.2)
## tidyverse      * 1.2.1      2017-11-14 [1] CRAN (R 3.5.2)
## usethis        1.4.0      2018-08-14 [1] CRAN (R 3.5.2)
## viridis        * 0.5.1      2018-03-29 [1] CRAN (R 3.5.2)
## viridisLite    * 0.3.0      2018-02-01 [1] CRAN (R 3.5.2)
## withr          2.1.2      2018-03-15 [1] CRAN (R 3.5.2)
## xfun           0.4        2018-10-23 [1] CRAN (R 3.5.2)
## xml2           1.2.0      2018-01-24 [1] CRAN (R 3.5.2)
## yaml          2.2.0      2018-07-25 [1] CRAN (R 3.5.2)
##
## [1] /home/travis/R/Library
## [2] /usr/local/lib/R/site-library
## [3] /home/travis/R-bin/lib/R/library
```

## References

Faraway, J. J. 2002. *Practical Regression and Anova using R*. [University of Bath]. Available at: <https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf> (<https://cran.r-project.org/doc/contrib/Faraway-PRA.pdf>).

Garnier, S. 2018. *Viridis: Default color maps from 'matplotlib'*. Available at: <https://CRAN.R-project.org/package=viridis> (<https://CRAN.R-project.org/package=viridis>).

R Core Team. 2017. *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. Available at: <https://www.R-project.org/> (<https://www.R-project.org/>).

Thompson, J. 2015. Modelling population densities of root-lesion nematode (*Pratylenchus thornei*) from soil profile temperatures to choose an optimum sowing date for wheat in a subtropical region. *Field Crops Research*. 183:50–55 Available at: <http://www.sciencedirect.com/science/article/pii/S0378429015300083> (<http://www.sciencedirect.com/science/article/pii/S0378429015300083>).

Wickham, H. 2016. *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York. Available at: <http://ggplot2.org> (<http://ggplot2.org>).

Wickham, H. 2014. Tidy data. *The Journal of Statistical Software*. 59 Available at: <http://www.jstatsoft.org/v59/i10/> (<http://www.jstatsoft.org/v59/i10/>).

Wickham, H. 2017. *tidyverse: Easily Install and Load the 'Tidyverse'*. Available at: <https://CRAN.R-project.org/package=tidyverse> (<https://CRAN.R-project.org/package=tidyverse>).

Wickham, H., Francois, R., Henry, L., and Müller, K. 2017. *dplyr: A Grammar of Data Manipulation*. Available at: <https://CRAN.R-project.org/package=dplyr> (<https://CRAN.R-project.org/package=dplyr>).

Wickham, H., and Henry, L. 2018. *tidyr: Easily Tidy Data with 'spread()' and 'gather()' Functions*. Available at: <https://CRAN.R-project.org/package=tidyr> (<https://CRAN.R-project.org/package=tidyr>).

Wickham, H., Hester, J., and Francois, R. 2017. *readr: Read Rectangular Text Data*. Available at: <https://CRAN.R-project.org/package=readr> (<https://CRAN.R-project.org/package=readr>).

