library(tidyverse)

## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.2 ✔ readr 2.1.4  
## ✔ forcats 1.0.0 ✔ stringr 1.5.0  
## ✔ ggplot2 3.4.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.2 ✔ tidyr 1.3.0  
## ✔ purrr 1.0.1   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 1.1.0 ──  
## ✔ broom 1.0.4 ✔ rsample 1.1.1  
## ✔ dials 1.2.0 ✔ tune 1.1.1  
## ✔ infer 1.0.4 ✔ workflows 1.1.3  
## ✔ modeldata 1.1.0 ✔ workflowsets 1.0.1  
## ✔ parsnip 1.1.0 ✔ yardstick 1.2.0  
## ✔ recipes 1.0.6   
## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## ✖ scales::discard() masks purrr::discard()  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ recipes::fixed() masks stringr::fixed()  
## ✖ dplyr::lag() masks stats::lag()  
## ✖ yardstick::spec() masks readr::spec()  
## ✖ recipes::step() masks stats::step()  
## • Dig deeper into tidy modeling with R at https://www.tmwr.org

library(GGally)

## Registered S3 method overwritten by 'GGally':  
## method from   
## +.gg ggplot2

library(lmtest)

## Loading required package: zoo  
##   
## Attaching package: 'zoo'  
##   
## The following objects are masked from 'package:base':  
##   
## as.Date, as.Date.numeric

air <- airquality

str(air)

## 'data.frame': 153 obs. of 6 variables:  
## $ Ozone : int 41 36 12 18 NA 28 23 19 8 NA ...  
## $ Solar.R: int 190 118 149 313 NA NA 299 99 19 194 ...  
## $ Wind : num 7.4 8 12.6 11.5 14.3 14.9 8.6 13.8 20.1 8.6 ...  
## $ Temp : int 67 72 74 62 56 66 65 59 61 69 ...  
## $ Month : int 5 5 5 5 5 5 5 5 5 5 ...  
## $ Day : int 1 2 3 4 5 6 7 8 9 10 ...

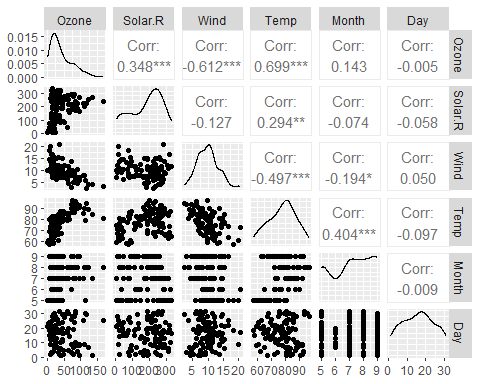
summary(air)

## Ozone Solar.R Wind Temp   
## Min. : 1.00 Min. : 7.0 Min. : 1.700 Min. :56.00   
## 1st Qu.: 18.00 1st Qu.:115.8 1st Qu.: 7.400 1st Qu.:72.00   
## Median : 31.50 Median :205.0 Median : 9.700 Median :79.00   
## Mean : 42.13 Mean :185.9 Mean : 9.958 Mean :77.88   
## 3rd Qu.: 63.25 3rd Qu.:258.8 3rd Qu.:11.500 3rd Qu.:85.00   
## Max. :168.00 Max. :334.0 Max. :20.700 Max. :97.00   
## NA's :37 NA's :7   
## Month Day   
## Min. :5.000 Min. : 1.0   
## 1st Qu.:6.000 1st Qu.: 8.0   
## Median :7.000 Median :16.0   
## Mean :6.993 Mean :15.8   
## 3rd Qu.:8.000 3rd Qu.:23.0   
## Max. :9.000 Max. :31.0   
##

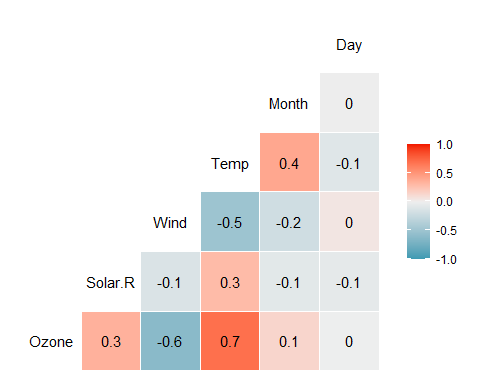
air2 <- drop\_na(air)  
str(air2)

## 'data.frame': 111 obs. of 6 variables:  
## $ Ozone : int 41 36 12 18 23 19 8 16 11 14 ...  
## $ Solar.R: int 190 118 149 313 299 99 19 256 290 274 ...  
## $ Wind : num 7.4 8 12.6 11.5 8.6 13.8 20.1 9.7 9.2 10.9 ...  
## $ Temp : int 67 72 74 62 65 59 61 69 66 68 ...  
## $ Month : int 5 5 5 5 5 5 5 5 5 5 ...  
## $ Day : int 1 2 3 4 7 8 9 12 13 14 ...

ggpairs(air2)



ggcorr(air2,label=TRUE)



Build model

lin\_reg = recipe(Ozone ~ Temp, air2)  
  
# Define the model  
lm\_model = linear\_reg()   
lm\_model = set\_engine(lm\_model, "lm")   
  
# Initiate the workflow  
lm\_wflow = workflow()  
  
# Add the model to the workflow  
lm\_wflow = add\_model(lm\_wflow, lm\_model)  
  
# Add the recipe to the workflow  
lm\_wflow = add\_recipe(lm\_wflow, lin\_reg)  
  
# Fit the model  
lm\_fit = fit(lm\_wflow, air2)  
  
summary(lm\_fit$fit$fit$fit)

##   
## Call:  
## stats::lm(formula = ..y ~ ., data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -40.922 -17.459 -0.874 10.444 118.078   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -147.6461 18.7553 -7.872 2.76e-12 \*\*\*  
## Temp 2.4391 0.2393 10.192 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 23.92 on 109 degrees of freedom  
## Multiple R-squared: 0.488, Adjusted R-squared: 0.4833   
## F-statistic: 103.9 on 1 and 109 DF, p-value: < 2.2e-16

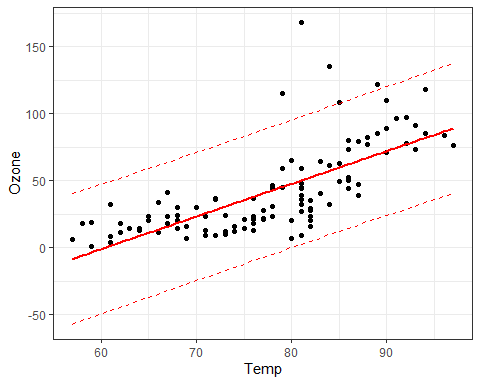
Examine prediction interval

#Prediction interval  
temp\_var = predict(lm\_fit$fit$fit$fit, interval = "prediction") #accessing the fit object with the three $fit

## Warning in predict.lm(lm\_fit$fit$fit$fit, interval = "prediction"): predictions on current data refer to \_future\_ responses

new\_df = cbind(air2, temp\_var)  
  
ggplot(new\_df, aes(x = Temp, y = Ozone)) +   
 geom\_point() +   
 geom\_smooth(method = "lm", se = FALSE, color = "red") +  
 geom\_line(aes(y=lwr), color = "red", linetype = "dashed") +  
 geom\_line(aes(y=upr), color = "red", linetype = "dashed") +  
 theme\_bw()

## `geom\_smooth()` using formula = 'y ~ x'

 Get lower and upper bound of slope at 95% confidence interval

# Assuming lm\_fit is your fitted model object  
  
# Load the required library  
library(broom)  
  
# Extract the model object from the workflow fit  
lm\_model\_fit = lm\_fit$fit$fit$fit  
  
# Obtain a tidy summary of your model  
tidy\_summary <- tidy(lm\_model\_fit)  
  
# Print the tidy summary  
print(tidy\_summary)

## # A tibble: 2 × 5  
## term estimate std.error statistic p.value  
## <chr> <dbl> <dbl> <dbl> <dbl>  
## 1 (Intercept) -148. 18.8 -7.87 2.76e-12  
## 2 Temp 2.44 0.239 10.2 1.55e-17

# To get the confidence interval for the slope you can do:  
confint(lm\_model\_fit, 'Temp', level = 0.95)

## 2.5 % 97.5 %  
## Temp 1.964787 2.913433

Predict Y with X

#Using predict function  
predict\_values = data.frame(Temp = c(80))  
predict(lm\_fit, new\_data = predict\_values)

## # A tibble: 1 × 1  
## .pred  
## <dbl>  
## 1 47.5

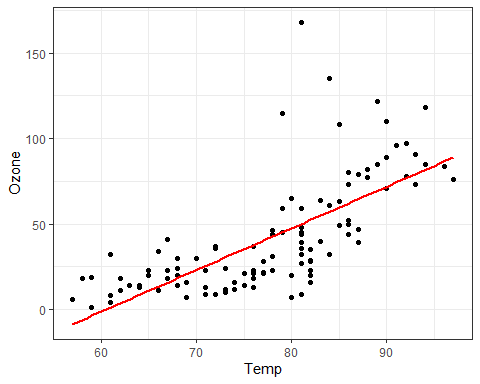
Diagnostics

Examine scatterplot to test Assumption 1:

**Assumption 1:** The predictor and response variable have a linear relationship

ggplot(air2, aes(x=Temp,y=Ozone)) + geom\_point() +   
 geom\_smooth(method="lm",se=FALSE, color="red") + theme\_bw()

## `geom\_smooth()` using formula = 'y ~ x'



Perform Durbin-Watson test to test independence of residuals for Assumption 2:

**Assumption 2:** Model errors (residuals) are independent

dwtest(lm\_fit$fit$fit$fit)

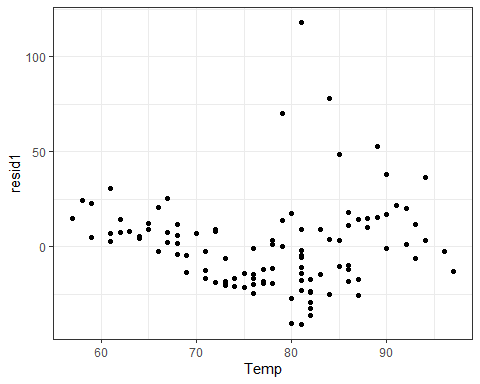
##   
## Durbin-Watson test  
##   
## data: lm\_fit$fit$fit$fit  
## DW = 1.8644, p-value = 0.2123  
## alternative hypothesis: true autocorrelation is greater than 0

We fail to reject the null hypothesis with a p-value greater than 0.05. This suggests that the residuals are likely independent.

Examine a plot of residual to test Assumption 3:

**Assumption 3:** Model residuals exhibit constant variance

air2 = mutate(air2, resid1 = lm\_fit$fit$fit$fit$residuals)  
ggplot(air2,aes(x=Temp,y=resid1)) + geom\_point() + theme\_bw()

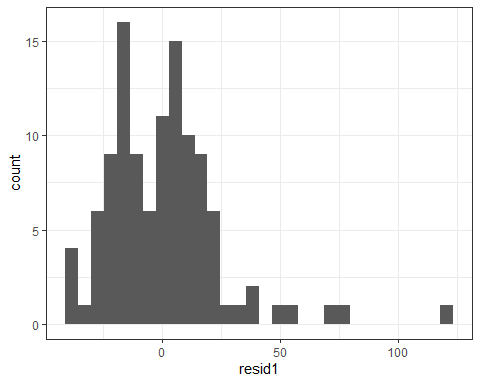
 Looks good, no clustering

Finally we examine a histogram of residuals to check Assumption 4:

**Assumption 4:** Model residuals are Normally-distributed

ggplot(air2,aes(x=resid1)) + geom\_histogram() + theme\_bw()

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



Kind of, I see a right skew though.