Linear regression, diagnostics, variable selection

2024-09-05

- **M1 MIDS & MFA**
- **Université Paris Cité**
- Année 2024-2025
- Course Homepage

- Moodle

! **Objectives**

Linear Regression on Whiteside data

Packages installation and loading (again)

We will use the following packages. If needed, we install them.

```
stopifnot(
  require(tidyverse),
                  require(broom),
                  require(magrittr),
                  require(lobstr),
                  require(ggforce),
#                  require(cowplot),
                  require(patchwork),
                  require(glue),
                  require(DT),
                  require(viridis)
)
```

Dataset

```
whiteside <- MASS::whiteside # no need to load the whole package

cur_dataset <- str_to_title(as.character(substitute(whiteside)))

# ?whiteside
```

> Mr Derek Whiteside of the UK Building Research Station recorded the weekly gas consumption and average external temperature at his own house in south-east England for two heating seasons, one of 26 weeks before, and one of 30 weeks after cavity-wall insulation was installed. The object of the exercise was to assess the effect of the insulation on gas consumption.

```
whiteside %>%
  glimpse
```

```
Rows: 56
Columns: 3
$ Insul <fct> Before, Before, Before, Before, Before, Before, Before, Before, ~
$ Temp  <dbl> -0.8, -0.7, 0.4, 2.5, 2.9, 3.2, 3.6, 3.9, 4.2, 4.3, 5.4, 6.0, 6.~
$ Gas   <dbl> 7.2, 6.9, 6.4, 6.0, 5.8, 5.8, 5.6, 4.7, 5.8, 5.2, 4.9, 4.9, 4.3,~
```

Start with columnwise and pairwise exploration

```
C <- whiteside %>%
  select(where(is.numeric)) %>%
  cov()

# Covariance between Gas and Temp

mu_n <- whiteside %>%
  select(where(is.numeric)) %>%
  colMeans()

# mu_n # Mean vector
```

$$C_n = \begin{bmatrix} 7.56 & -2.19 \\ -2.19 & 1.36 \end{bmatrix} \qquad \mu_n = \begin{bmatrix} 4.88 \\ 4.07 \end{bmatrix}$$

Use `skimr::skim()` to write univariate reports

Solution

```
sk <- whiteside %>%
  skimr::skim() %>%
  select(-n_missing, - complete_rate)

skimr::yank(sk, "factor")
```

Variable type: factor

skim_variable	ordered	n_unique	top_counts
Insul	FALSE	2	Aft: 30, Bef: 26

```
skimr::yank(sk, "numeric")
```

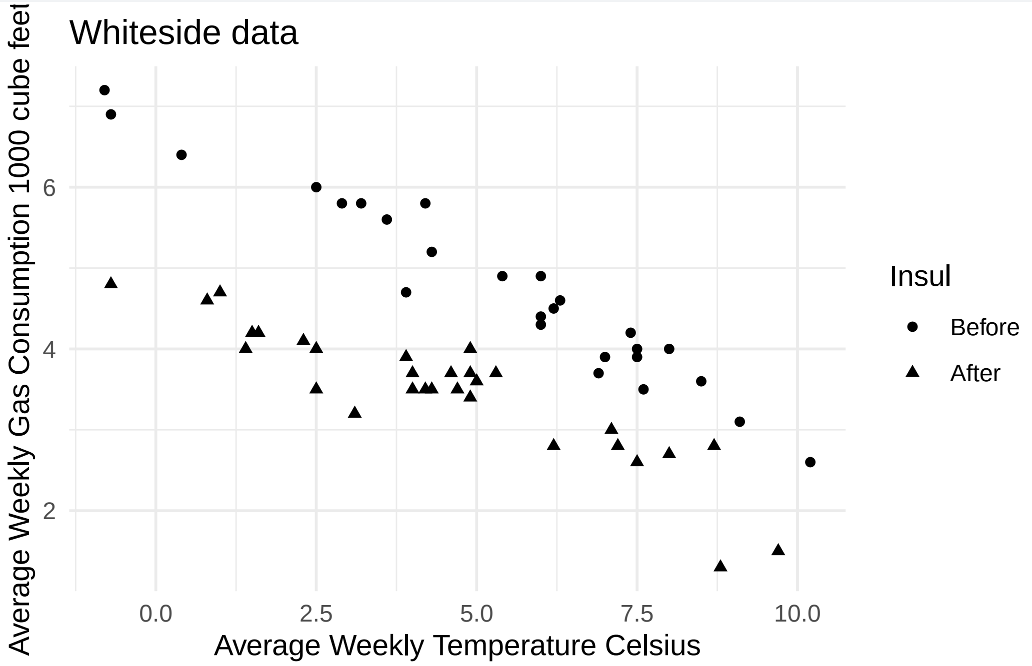
Variable type: numeric

skim_variable	mean	sd	p0	p25	p50	p75	p100	hist
Temp	4.88	2.75	-0.8	3.05	4.90	7.12	10.2	
Gas	4.07	1.17	1.3	3.50	3.95	4.62	7.2	

Build a scatterplot of the Whiteside dataset

Solution

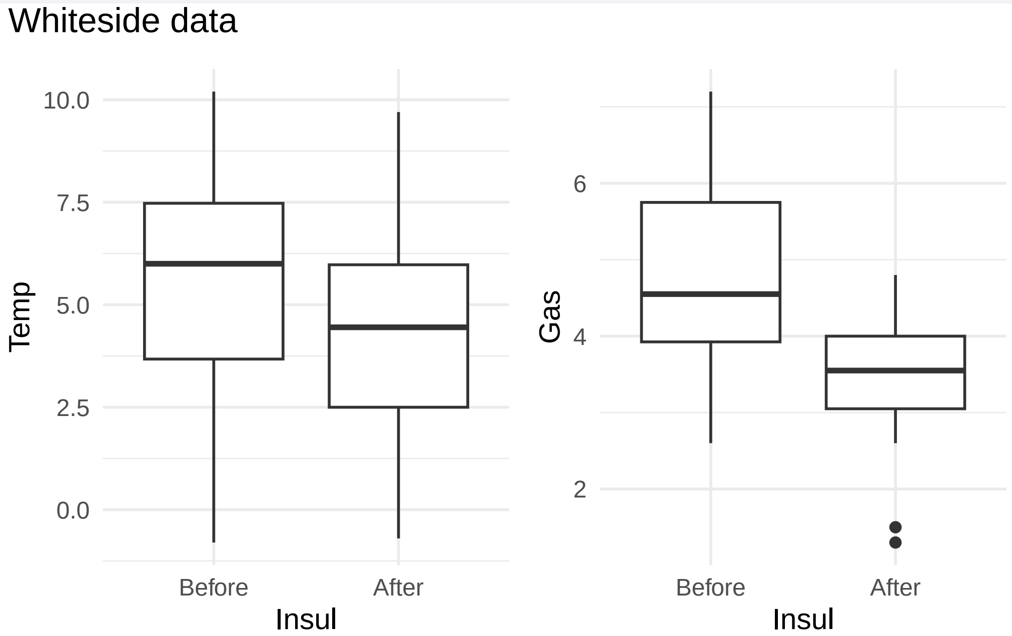
```
p <- whiteside %>%
  ggplot() +
  aes(x=Temp, y=Gas) +
  geom_point(aes(shape=Insul)) +
  xlab("Average Weekly Temperature Celsius") +
  ylab("Average Weekly Gas Consumption 1000 cube feet")

p +
  ggtitle(glue("{cur_dataset} data"))
```



Build boxplots of `Temp` and `Gas` versus `Insul`

Solution

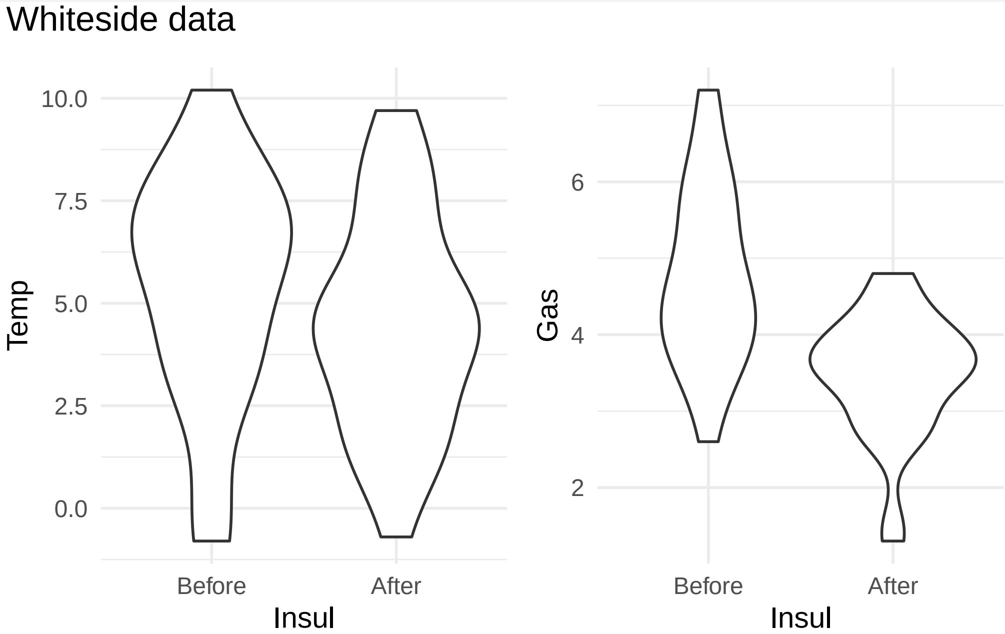
```r
q <- whiteside %>%
  ggplot() +
  aes(x=Insul)

qt <- q +
  geom_boxplot(aes(y=Temp))

qg <- q +
  geom_boxplot(aes(y=Gas))

(qt + qg) +
  patchwork::plot_annotation(title = glue("{cur_dataset} data"))
```



Build violine plots of `Temp` and `Gas` versus `Insul`

Solution

```
(q +
  geom_violin(aes(y=Temp))) +
(q +
  geom_violin(aes(y=Gas))) +
  patchwork::plot_annotation(title = glue("{cur_dataset} data"))
```

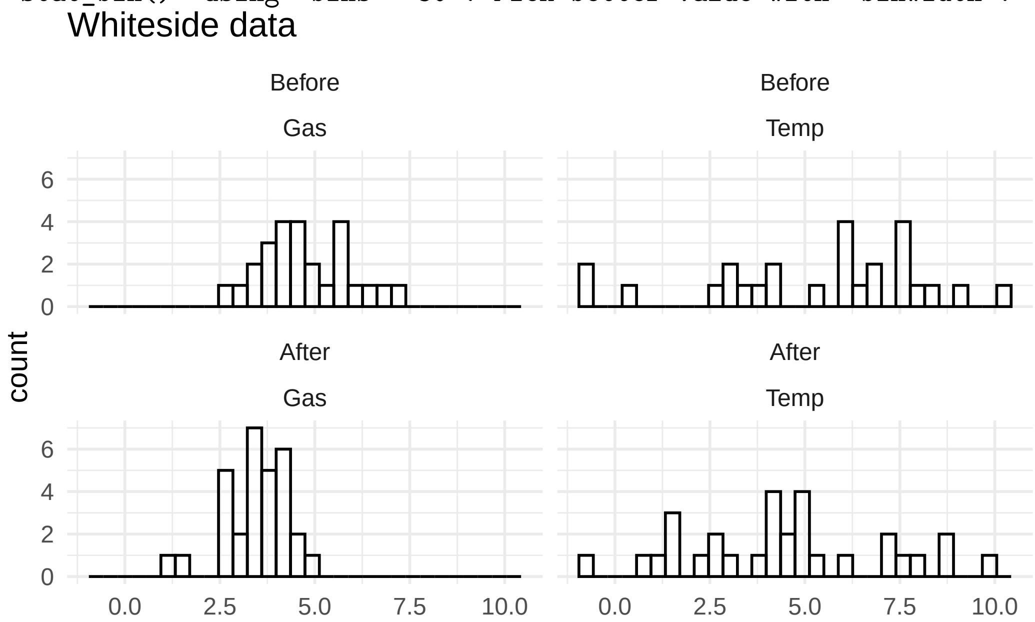


Plot histograms of `Temp` and `Gas` versus `Insul`

Solution

```
r <- whiteside %>%
  pivot_longer(cols=c(Gas, Temp),
               names_to = "Vars",
               values_to = "Vals") %>%
  ggplot() +
  aes(x=Vals)  +
  facet_wrap(~ Insul + Vars ) +
  xlab("")

r +
  geom_histogram(alpha=.3, fill="white", color="black") +
  ggtitle(glue("{cur_dataset} data"))
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

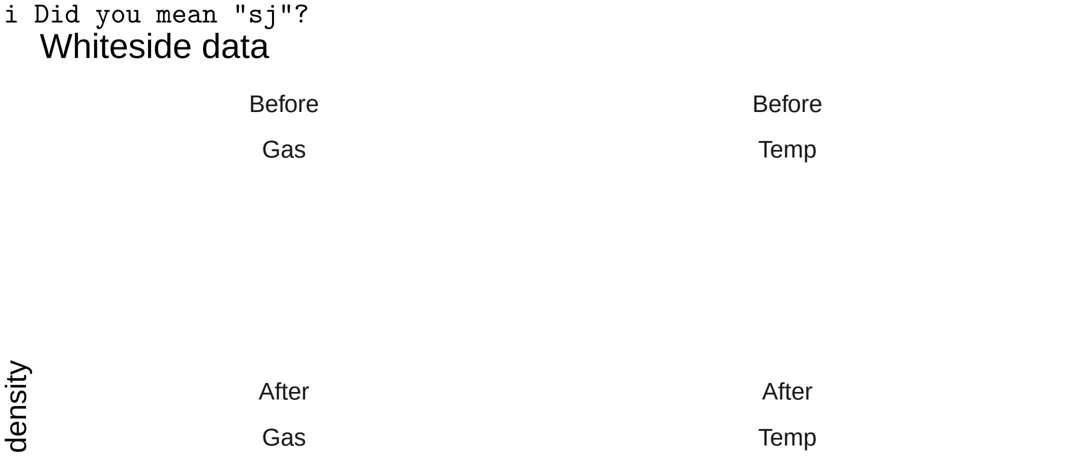


Plot density estimates of `Temp` and `Gas` versus `Insul`.

Solution

```r
r +
  stat_density(alpha=.3 ,
               fill="white",
               color="black",
               bw = "SJ",
               adjust = .5 ) +
  ggtitle(glue("{cur_dataset} data"))
```

```
Warning: Computation failed in `stat_density()`.
Computation failed in `stat_density()`.
Computation failed in `stat_density()`.
Computation failed in `stat_density()`.
Caused by error in `precompute_bw()`:
! `bw` must be one of "nrd0", "nrd", "ucv", "bcv", "sj", "sj-ste", or
  "sj-dpi", not "SJ".
i Did you mean "sj"?
```



Hand-made calculatoin of simple linear regression estimates for `Gas` versus `Temp`

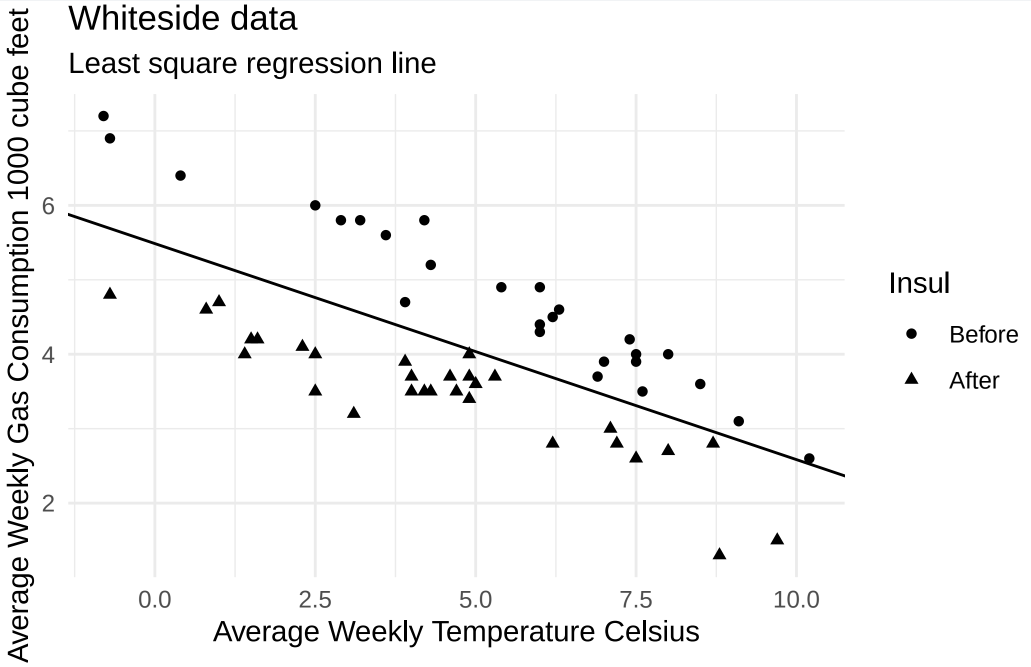
Solution

```r
b <- C[1,2] / C[1,1] # slope

a <- whiteside %$%  # exposing pipe from magrittr
  (mean(Gas) - b * mean(Temp)) # intercept

# with(whiteside,
#     mean(Gas) - b * mean(Temp))
```

Overlay the scatterplot with the regression line.

Solution

```r
p +
  geom_abline(slope=b, intercept = a) +
  ggtitle(glue("{cur_dataset} data"), subtitle = "Least square regression line")
```



Using `lm()`

`lm` stands for Linear Models. Function `lm` has a number of arguments, including:

- formula
- data

Solution

```
lm0 <- lm(Gas ~ Temp, data = whiteside)
```

The result is an object of class `lm`. The generic function `summary()` has a method for class `lm`

```
lm0 %>%
  summary()
```

```
Call:
lm(formula = Gas ~ Temp, data = whiteside)

Residuals:
    Min      1Q  Median      3Q     Max
-1.6324 -0.7119 -0.2047  0.8187  1.5327

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   5.4862     0.2357  23.275  < 2e-16 ***
Temp         -0.2902     0.0422  -6.876 6.55e-09 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.8606 on 54 degrees of freedom
Multiple R-squared:  0.4668,    Adjusted R-squared:  0.457
F-statistic: 47.28 on 1 and 54 DF,  p-value: 6.545e-09
```

The summary is made of four parts

- The call. Very useful if we handle many different models (corresponding to different formulae, or different datasets)
- A numerical summary of residuals
- A commented display of the estimated coefficients
- Estimate of noise scale (under Gaussian assumptions)
- Squared linear correlation coefficient between response variable Y (`Gas`) and predictions $\widehat{Y}$
- A test statistic (Fisher's statistic) for assessing null hypothesis that slope is null, and corresponding p-value (under Gaussian assumptions).

Including a rough summary in a report is not always a good idea. It is easy to extract a tabular version of the summary using functions `tidy()` and `glance()` from package `broom`.

For html output `DT::datatable()` allows us to polish the final output

Solution

```r
tidy_lm <-  . %$% (
  tidy(.) %>%
  mutate(across(-c(term, p.value), \(x) round(x, digits=2)),
         p.value = signif(p.value, 3)) %>%
  DT::datatable(extensions="Responsive",
                options = list(dom = 't'),
                caption = glue("Dataset {call$data},  {deparse(call$formula)}"))
)

tidy_lm(lm0)
```

```
PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is ir
```

Dataset whiteside, Gas ~ Temp

	term	estimate	std.error	statistic	p.value
1	(Intercept)	5.49	0.24	23.28	7.97e-30
2	Temp	-0.29	0.04	-6.88	6.55e-9

Function `glance()` extract informations that can be helpful when performing model/variable selection.

Solution

```r
glance_lm <-  . %$% (
  glance(.) %>%
  mutate(across(-c(p.value),
                ~ round(.x, digits=2)),
         p.value=signif(p.value,3)) %>%
  DT::datatable(extensions="Responsive",
                options = list(dom = 't'),
                caption = glue("Dataset {call$data},  {deparse(call$formula)}"))
)

glance_lm(lm0)
```

Dataset whiteside, Gas ~ Temp

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.47	0.46	0.86	47.28	6.55e-9	1	-70.04	146.07	152.15	39.99	54	56

`R` offers a function `confint()` that can be fed with objects of class `lm`. Explain the output of this function.

Solution

```r
confint(lm0, level=.99)
```

```
                 0.5 %     99.5 %
(Intercept)  4.8568584  6.1155283
Temp        -0.4028939 -0.1775224
```

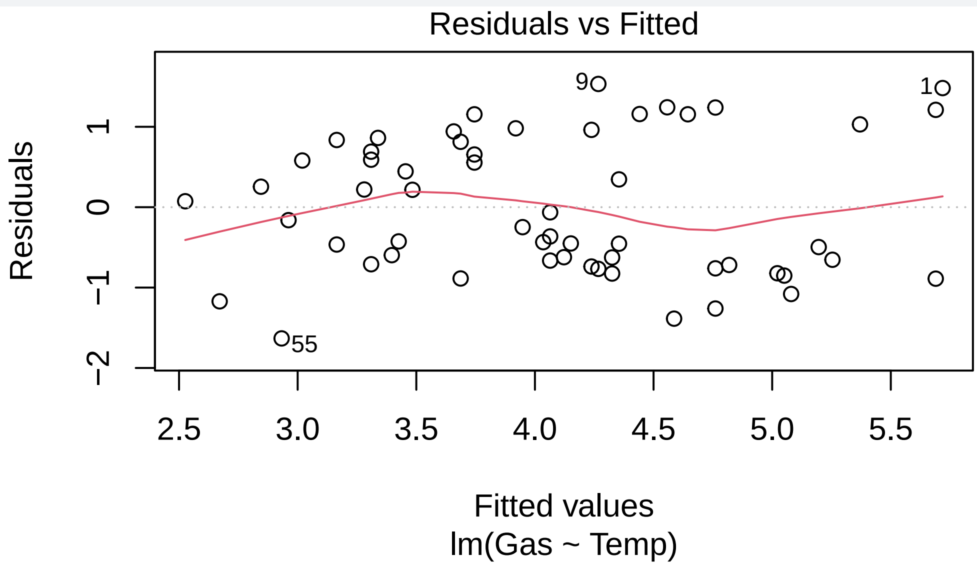
Diagnostic plots

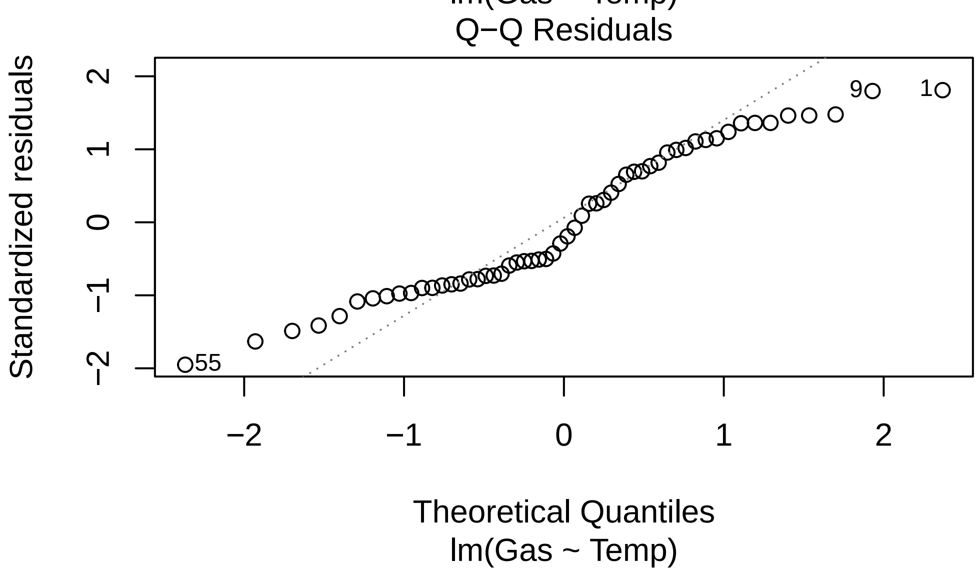
Method `plot.lm()` of generic S3 function `plot` from base `R` offers six diagnostic plots. By default it displays four of them.

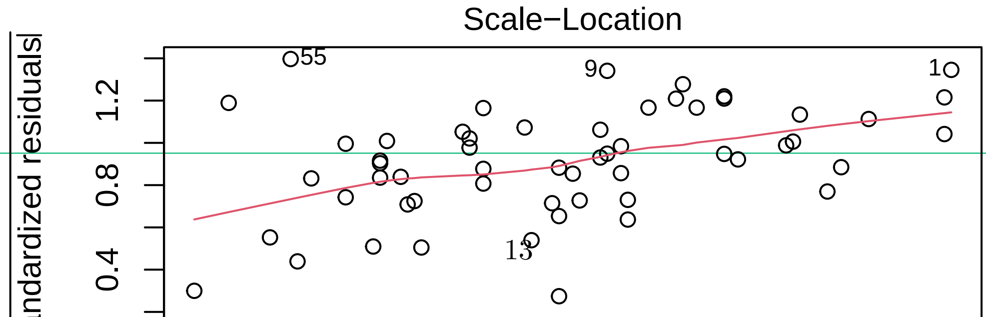
What are the diagnostic plots good for?

Solution

```
plot(lm0)
```







These diagnostic plots can be built from the information gathered in the `lm` object returned by `lm(...)`.

It is convenient to extract the required pieces of information using method `augment.lm`. of *generic function* `augment()` from package `broom`.

Solution

```
whiteside_aug <- lm0 %>%
  augment(whiteside)

lm0 %$% ( # exposing pipe !!!
  augment(., data=whiteside) %>%
  mutate(across(!where(is.factor), ~ signif(.x, 3))) %>%
  group_by(Insul) %>%
  sample_n(5) %>%
  ungroup() %>%
   DT::datatable(extensions="Responsive",
                 caption = glue("Dataset {call$data},  {deparse(call$formula)}"))
)
```

Show 10 entries Search:

Dataset whiteside, Gas ~ Temp

	Insul	Temp	Gas	.fitted	.resid	.hat	.sigma	.cooksd	.std.resid
1	Before	6.9	3.7	3.48	0.216	0.0277	0.868	0.000926	0.255
2	Before	6.3	4.6	3.66	0.942	0.0227	0.859	0.0143	1.11
3	Before	-0.8	7.2	5.72	1.48	0.0953	0.842	0.173	1.81
4	Before	10.2	2.6	2.53	0.0739	0.0861	0.869	0.00038	0.0899
5	Before	6	4.3	3.74	0.555	0.0209	0.865	0.00453	0.652
6	After	4.2	3.5	4.27	-0.767	0.019	0.862	0.00783	-0.9
7	After	4.9	3.7	4.06	-0.364	0.0179	0.867	0.00166	-0.427
8	After	4.6	3.7	4.15	-0.451	0.018	0.866	0.00257	-0.529
9	After	1.5	4.2	5.05	-0.851	0.0453	0.86	0.0243	-1.01
10	After	4.7	3.5	4.12	-0.622	0.0179	0.864	0.00486	-0.73

Showing 1 to 10 of 10 entries Previous 1 Next

Recall that in the output of `augment()`

- `.fitted`: $\widehat{Y} = H \times Y = X \times \hat{\beta}$
- `.resid`: $\hat{\epsilon} = Y - \widehat{Y}$ residuals, $\sim (\text{Id}_n - H) \times \epsilon$
- `.hat`: diagonal coefficients of Hat matrix H
- `.sigma`: is meant to be the estimated standard deviation of components of $\widehat{Y}$

Compute the share of *explained variance*

Solution

```
whiteside_aug %$% {
  1 - (var(.resid)/(var(Gas)))
}
```

```
[1] 0.4668366
```

```
# with(whiteside_aug,
#   1 - (var(.resid)/(var(Gas)))
```

Plot residuals against fitted values

Solution

```
diag_1 <- whiteside_aug %>%
  ggplot() +
  aes(x=.fitted, y=.resid)+
  geom_point(aes(shape= Insul), size=1, color="black") +
  geom_smooth(formula = y ~ x,
              method="loess",
              se=F,
              linetype="dotted",
              linewidth=.5,
              color="black") +
  geom_hline(yintercept = 0, linetype="dashed") +
  xlab("Fitted values") +
  ylab("Residuals)") +
  labs(caption = "Residuals versus Fitted")
```

Fitted against square root of standardized residuals.

Solution

```
diag_3 <- whiteside_aug %>%
  ggplot() +
  aes(x=.fitted, y=sqrt(abs(.std.resid))) +
  geom_smooth(formula = y ~ x,
              se=F,
              method="loess",
              linetype="dotted",
              linewidth=.5,
              color="black") +
  xlab("Fitted values") +
  ylab("sqrt(|standardized residuals|)") +
  geom_point(aes(shape=Insul), size=1, alpha=1) +
  labs(caption = "Scale location")
```

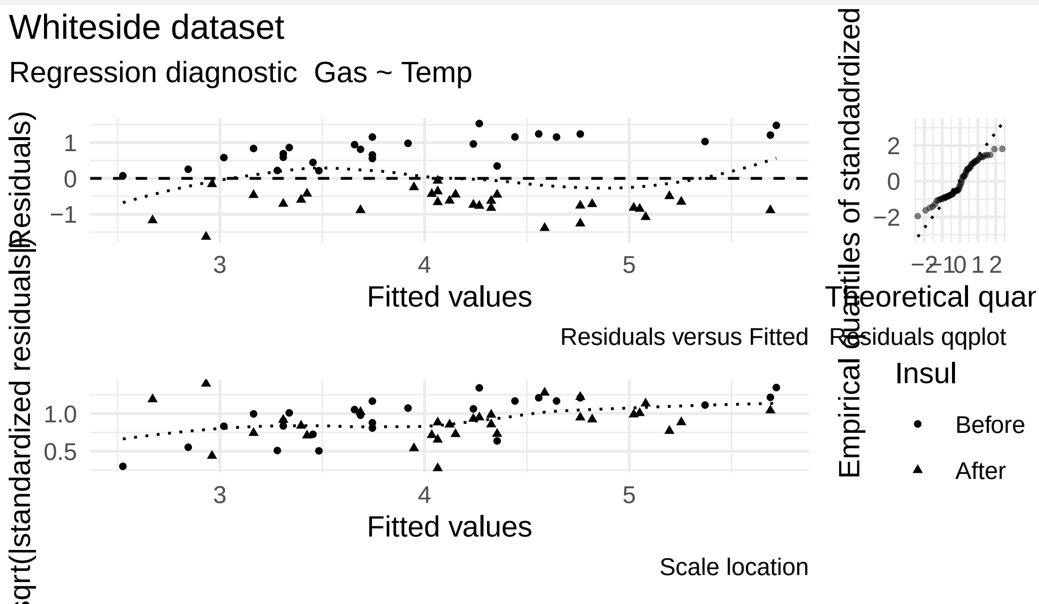
Solution

```
diag_2 <- whiteside_aug %>%
  ggplot() +
  aes(sample=.std.resid) +
  geom_qq(size=.5, alpha=.5) +
  stat_qq_line(linetype="dotted",
               linewidth=.5,
               color="black") +
  coord_fixed() +
  labs(caption="Residuals qqplot") +
  xlab("Theoretical quantiles") +
  ylab("Empirical quantiles of standadrdized residuals")
```

TAF

Solution

```
(diag_1 + diag_2 + diag_3 + guide_area()) +
  plot_layout(guides="collect") +
  plot_annotation(title=glue("{cur_dataset} dataset"),
                  subtitle = glue("Regression diagnostic  {deparse(lm0$call$formula)}"
                  )
```



The fact that the sign of residuals depend on Insul shows that our modeling is too naive.
The qqplot suggests that the residuals are not collected from Gaussian homoschedastic noise.

Taking into account Insulation

Design a *formula* that allows us to take into account the possible impact of Insulation. Insulation may impact the relation between weekly `Gas` consumption and average external `Temperature` in two ways. Insulation may modify the `Intercept`, it may also modify the slope, that is the sensitivity of `Gas` consumption with respect to average external `Temperature`.

Have a look at formula documentation (`?formula`).

Solution

```
lm1 <- lm(Gas ~ Temp * Insul, data = whiteside)
```

Check the design using function `model.matrix()`. How can you relate this augmented design and the *one-hot encoding* of variable `Insul`?

Solution

```
model.matrix(lm1) |>
  head()
```

```
  (Intercept) Temp InsulAfter Temp:InsulAfter
1           1 -0.8          0               0
2           1 -0.7          0               0
3           1  0.4          0               0
4           1  2.5          0               0
5           1  2.9          0               0
6           1  3.2          0               0
```

Solution

```
lm1 %>%
  tidy_lm()
```

Dataset whiteside, Gas ~ Temp * Insul

	term	estimate	std.error	statistic	p.value
1	(Intercept)	6.85	0.14	50.41	8e-46
2	Temp	-0.39	0.02	-17.49	1.98e-23
3	InsulAfter	-2.13	0.18	-11.83	2.32e-16
4	Temp:InsulAfter	0.12	0.03	3.59	0.000731

Solution

```
lm0 |>
  glance_lm()
```

Dataset whiteside, Gas ~ Temp

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.47	0.46	0.86	47.28	6.55e-9	1	-70.04	146.07	152.15	39.99	54	56

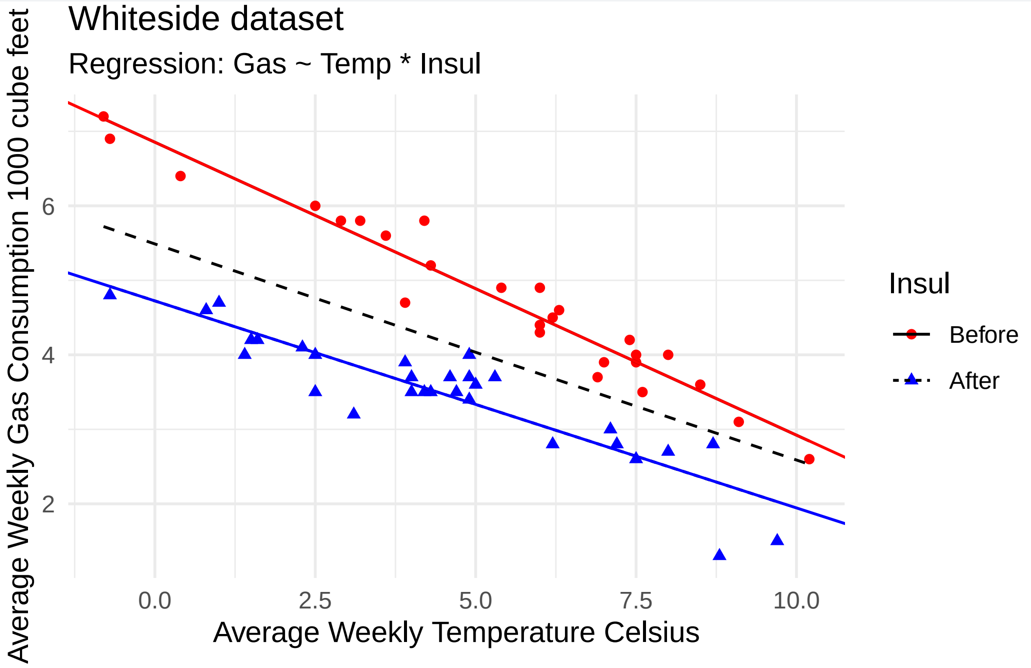
```
lm1 %>%
  glance_lm()
```

Dataset whiteside, Gas ~ Temp * Insul

	r.squared	adj.r.squared	sigma	statistic	p.value	df	logLik	AIC	BIC	deviance	df.residual	nobs
1	0.93	0.92	0.32	222.33	1.23e-29	3	-14.1	38.2	48.33	5.43	52	56

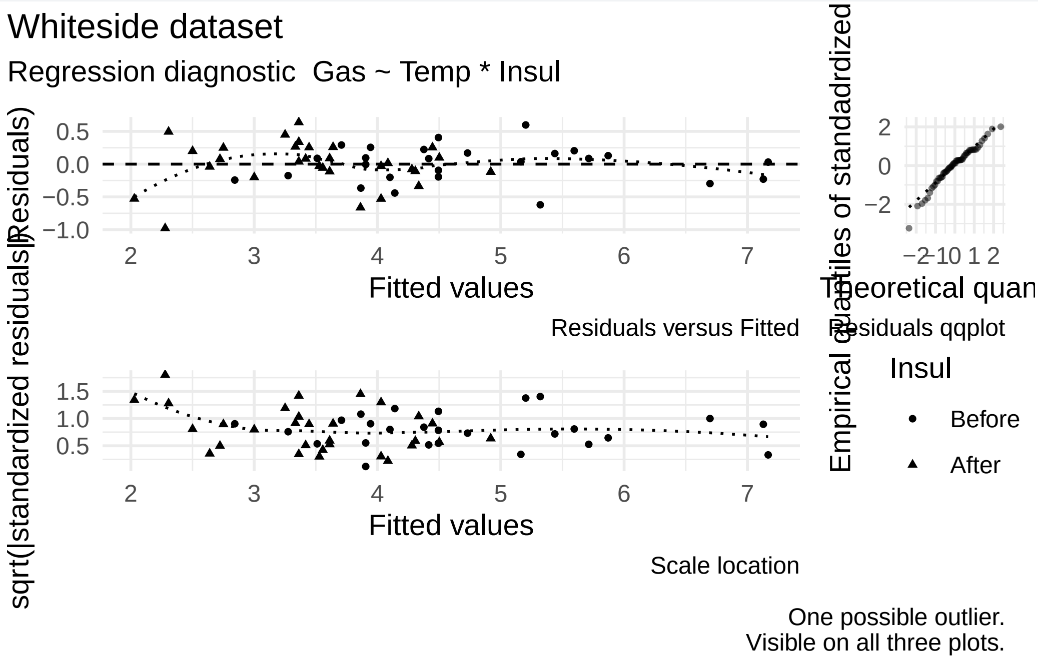
Solution

```
p +
  geom_smooth(formula='y ~ poly(x, 2)',linewidth=.5, color="black",linetype="dashed",
  aes(color=Insul) +
  geom_smooth(aes(linetype=Insul),
              formula='y ~ x',linewidth=.5, color="black", method="lm", se=FALSE) +
  scale_color_manual(values= c("Before"="red", "After"="blue")) +
  geom_abline(intercept = 6.8538, slope=-.3932, color="red") +
  geom_abline(intercept = 6.8538 - 2.13, slope=-.3932 +.1153, color="blue") + labs(
    title=glue("{cur_dataset} dataset"),
    subtitle = glue("Regression: {deparse(lm1$call$formula)}")
    )
```



Solution

```
whiteside_aug1 <-  augment(lm1, whiteside)

(diag_1 %+% whiteside_aug1) +
(diag_2 %+% whiteside_aug1) +
(diag_3 %+% whiteside_aug1) +
 guide_area() +
  plot_layout(guides = "collect") +
  plot_annotation(title=glue("{cur_dataset} dataset"),
                  subtitle = glue("Regression diagnostic  {deparse(lm1$call$formula)}'
                  )
```



The formula argument defines the design matrix and the Least-Squares problem used to estimate the coefficients.

Function `model.matrix()` allows us to inspect the design matrix.

Solution

```
model.matrix(lm1) %>%
  as_tibble() %>%
  mutate(Insul=ifelse(InsulAfter,"After", "Before")) %>%
  ungroup() %>%
  DT::datatable(caption=glue("Design matrix for {deparse(lm1$call$formula)}"))
```

Show 10 entries Search:

Design matrix for Gas ~ Temp * Insul

	(Intercept)	Temp	InsulAfter	Temp:InsulAfter	Insul
1	1	-0.8	0	0	Before
2	1	-0.7	0	0	Before
3	1	0.4	0	0	Before
4	1	2.5	0	0	Before
5	1	2.9	0	0	Before
6	1	3.2	0	0	Before
7	1	3.6	0	0	Before
8	1	3.9	0	0	Before
9	1	4.2	0	0	Before
10	1	4.3	0	0	Before

Showing 1 to 10 of 56 entries Previous 1 2 3 4 5 6 Next

```
X <- model.matrix(lm1)
```

In order to solve le Least-Square problems, we have to compute

$$(X^T \times X)^{-1} \times X^T$$

This can be done in several ways.

`lm()` uses QR factorization.

Solution

```
Q <- qr.Q(lm1$qr)
R <- qr.R(lm1$qr)  # R is upper triangular

norm(X - Q %*% R, type="F") # QR Factorization
```

```
[1] 1.753321e-14
```

```
signif(t(Q) %*% Q, 2)       # Q's columns form an orthonormal family
```

```
         [,1]     [,2]     [,3]    [,4]
[1,]  1.0e+00 -1.4e-17  3.1e-17 1.7e-16
[2,] -1.4e-17  1.0e+00 -3.5e-17 1.4e-17
[3,]  3.1e-17 -3.5e-17  1.0e+00 0.0e+00
[4,]  1.7e-16  1.4e-17  0.0e+00 1.0e+00
```

```
H <- Q %*% t(Q)              # The Hat matrix

norm(X - H %*% X, type="F") # H leaves X's columns invariant
```

```
[1] 1.758479e-14
```

```
norm(H - H %*% H, type="F") # H is idempotent
```

```
[1] 7.993681e-16
```

```
# eigen(H, symmetric = TRUE, only.values = TRUE)$values
```

```
sum((solve(t(X) %*% X) %*% t(X) %*% whiteside$Gas - lm1$coefficients)^2)
```

```
[1] 3.075652e-29
```

Once we have the QR factorization of X, solving the normal equations boils down to inverting a triangular matrix.

```
sum((solve(R) %*% t(Q) %*% whiteside$Gas - lm1$coefficients)^2)
```

```
[1] 2.050287e-29
```

```
#matador::mat2latex(signif(solve(t(X) %*% X), 2))
```

$$(X^T \times X)^{-1} = \begin{bmatrix} 0.18 & -0.026 & -0.18 & 0.026 \\ -0.026 & 0.0048 & 0.026 & -0.0048 \\ -0.18 & 0.026 & 0.31 & -0.048 \\ 0.026 & -0.0048 & -0.048 & 0.0099 \end{bmatrix}$$

Solution

```r
whiteside_aug1 %>%
  glimpse()
```

```
Rows: 56
Columns: 9
$ Insul      <fct> Before, Before, Before, Before, Before, Before, Before, Bef~
$ Temp       <dbl> -0.8, -0.7, 0.4, 2.5, 2.9, 3.2, 3.6, 3.9, 4.2, 4.3, 5.4, 6.~
$ Gas        <dbl> 7.2, 6.9, 6.4, 6.0, 5.8, 5.8, 5.6, 4.7, 5.8, 5.2, 4.9, 4.9,~
$ .fitted    <dbl> 7.168419, 7.129095, 6.696532, 5.870731, 5.713435, 5.595463,~
$ .resid     <dbl> 0.031581243, -0.229094875, -0.296532170, 0.129269357, 0.086~
$ .hat       <dbl> 0.22177670, 0.21586370, 0.15721835, 0.07782904, 0.06755399,~
$ .sigma     <dbl> 0.3261170, 0.3241373, 0.3230041, 0.3256103, 0.3259138, 0.32~
$ .cooksd    <dbl> 0.0008751645, 0.0441520664, 0.0466380672, 0.0036646607, 0.0~
$ .std.resid <dbl> 0.11083298, -0.80096122, -1.00001423, 0.41675591, 0.2775375~
```

Understanding `.fitted` column

Solution

```r
sum((predict(lm1, newdata = whiteside) - whiteside_aug1$.fitted)^2)
```

```
[1] 0
```

```r
sum((H %*% whiteside_aug1$Gas - whiteside_aug1$.fitted)^2)
```

```
[1] 3.478877e-28
```

Understanding `.resid`

Solution

```r
sum((whiteside_aug1$.resid + H %*% whiteside_aug1$Gas - whiteside_aug1$Gas)^2)
```

```
[1] 3.461127e-28
```

Understanding `.hat`

Solution

```r
sum((whiteside_aug1$.hat - diag(H))^2)
```

```
[1] 0
```

Understanding `.std.resid`

Solution

```
sigma_hat <- sqrt(sum(lm1$residuals^2)/lm1$df.residual)

lm1 %>% glance()
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
      <dbl>         <dbl> <dbl>     <dbl>    <dbl> <dbl>  <dbl> <dbl> <dbl>
1     0.928         0.924 0.323      222.  1.23e-29     3  -14.1  38.2  48.3
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

$$\hat{r}_i = \frac{\widehat{\epsilon}_i}{\widehat{\sigma}\sqrt{1 - H_{i,i}}}$$

```
sum((sigma_hat * sqrt(1 -whiteside_aug1$.hat) * whiteside_aug1$.std.resid - whiteside_
```

```
[1] 4.471837e-28
```

Understanding column `.sigma`

Solution

Column `.sigma` contains the *leave-one-out* estimates of σ, that is `whiteside_aug1$.sigma[i]` is the estimate of σ you obtain by leaving out the `i` row of the dataframe.
There is no need to recompute everything for each sample element.

$$\widehat{\sigma}^2_{(i)} = \widehat{\sigma}^2 \frac{n - p - 1 - \frac{\widehat{\epsilon}_i^2}{\widehat{\sigma}^2(1 - H_{i,i})}}{n - p - 2}$$