Lab05 Write-Up

Dohyun Leee

Part 1: COVIDCAST Data

```
covid = read_csv("covid_data.csv")
```

```
## Parsed with column specification:
## cols(
##   signal = col_character(),
##   geo_value = col_character(),
##   time_value = col_date(format = ""),
##   value = col_double(),
##   stderr = col_double(),
##   sample_size = col_double()
## )
```

```
covid
```

```
## # A tibble: 65,688 x 6
##    signal       geo_value time_value value stderr sample_size
##    <chr>        <chr>     <date>     <dbl>  <dbl>       <dbl>
##  1 smoothed_cli ak        2020-10-01 1.59  0.429         760
##  2 smoothed_cli al        2020-10-01 0.983 0.141        3742.
##  3 smoothed_cli ar        2020-10-01 1.06  0.174        2587.
##  4 smoothed_cli az        2020-10-01 0.597 0.0905       5682.
##  5 smoothed_cli ca        2020-10-01 0.450 0.0399      21930.
##  6 smoothed_cli co        2020-10-01 0.561 0.0917       5137.
##  7 smoothed_cli ct        2020-10-01 0.444 0.0978       3866
##  8 smoothed_cli dc        2020-10-01 0.268 0.224         533
##  9 smoothed_cli de        2020-10-01 0.203 0.134        1129.
## 10 smoothed_cli fl        2020-10-01 0.516 0.0472      17767.
## # ... with 65,678 more rows
```

```
covid %>%
  group_by(geo_value, signal) %>%
  summarize(
    avg = mean(value, na.rm = T)
  ) %>%
  pivot_wider(id_cols = geo_value, names_from = signal, values_from = avg) %>%
  ungroup() -> state_avg
```

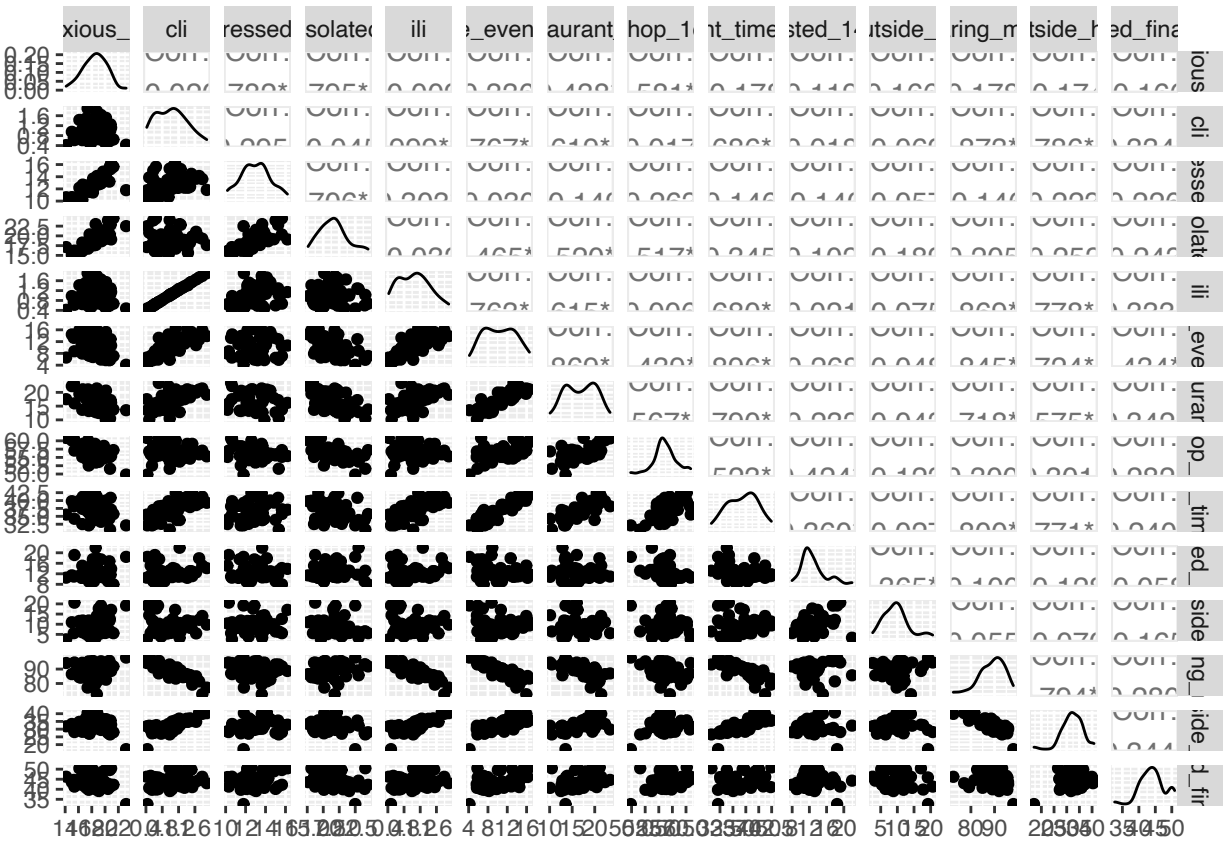
```
## `summarise()` has grouped output by 'geo_value'. You can override using the `.groups` argume
```

```
state_avg
```

```
## # A tibble: 51 x 15
##    geo_value smoothed_anxiou~ smoothed_cli smoothed_depres~ smoothed_felt_i~
##    <chr>                <dbl>        <dbl>            <dbl>            <dbl>
##  1 ak                    18.6        1.22             13.7             23.0
##  2 al                    16.6        1.41             12.7             16.7
##  3 ar                    19.7        1.20             14.8             19.1
##  4 az                    16.6        1.05             12.0             18.3
##  5 ca                    17.7        0.774            12.3             20.4
##  6 co                    20.0        1.00             13.7             21.6
##  7 ct                    18.2        0.723            11.4             17.8
##  8 dc                    23.2        0.434            11.8             22.5
##  9 de                    15.4        0.618            10.3             16.2
## 10 fl                    15.3        0.715            10.5             15.9
## # ... with 41 more rows, and 10 more variables: smoothed_ili <dbl>,
## #   smoothed_large_event_1d <dbl>, smoothed_restaurant_1d <dbl>,
## #   smoothed_shop_1d <dbl>, smoothed_spent_time_1d <dbl>,
## #   smoothed_tested_14d <dbl>, smoothed_travel_outside_state_5d <dbl>,
## #   smoothed_wearing_mask <dbl>, smoothed_work_outside_home_1d <dbl>,
## #   smoothed_worried_finances <dbl>
```

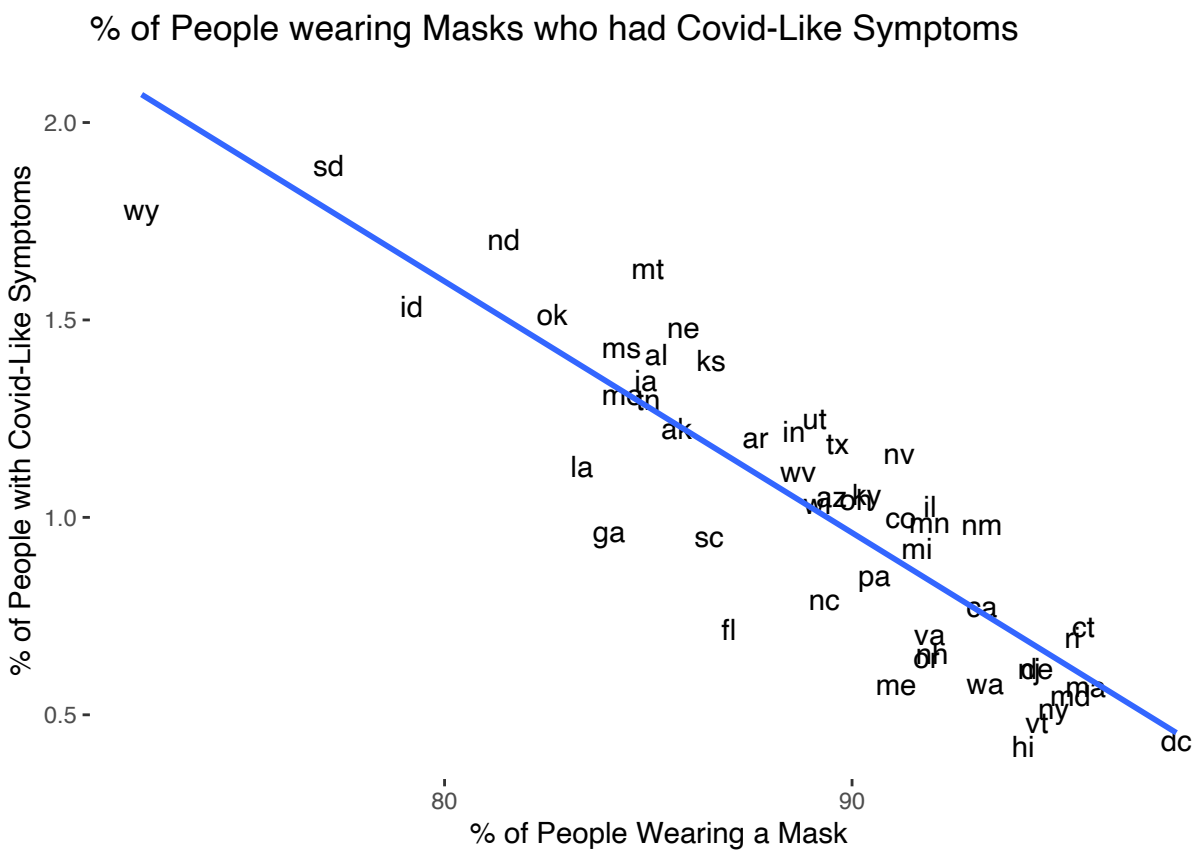
```
state_avg %>%
  select(-geo_value) %>%
  rename_all(funs(stringr::str_replace(., "smoothed_", ""))) %>%
  GGally::ggpairs()
```

```
## Warning: `funs()` is deprecated as of dplyr 0.8.0.
## Please use a list of either functions or lambdas:
##
##   # Simple named list:
##   list(mean = mean, median = median)
##
##   # Auto named with `tibble::lst()`:
##   tibble::lst(mean, median)
##
##   # Using lambdas
##   list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```




```
state_avg %>%
  mutate(state = str_to_upper(geo_value)) %>%
  ggplot(aes(x = smoothed_wearing_mask, y = smoothed_cli)) +
  geom_text(aes(label = geo_value)) +
  stat_smooth(method = "lm", se = FALSE) +
  labs(title = "% of People wearing Masks who had Covid-Like Symptoms",
       x = "% of People Wearing a Mask",
       y = "% of People with Covid-Like Symptoms") +
  theme(panel.background = element_blank())
```

```
## `geom_smooth()` using formula 'y ~ x'
```



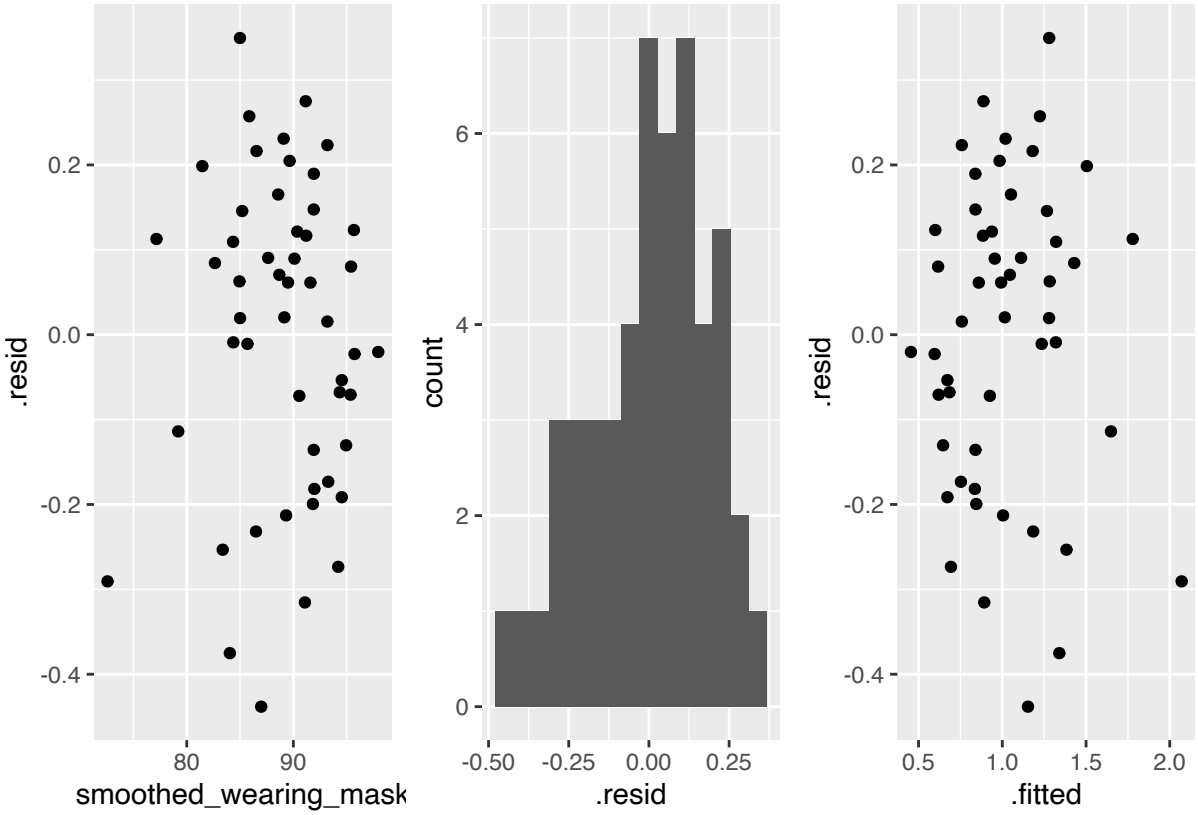

```
lm_mod = lm(smoothed_cli ~ smoothed_wearing_mask, data = state_avg)

lm_res = augment(lm_mod, interval = "prediction")
p1 = ggplot(lm_res, aes(x = smoothed_wearing_mask, y = .resid)) +
  geom_point()

p2 = ggplot(lm_res, aes(x = .resid)) +
  geom_histogram(bins = 15)

p3 = ggplot(lm_res, aes(x = .fitted, y = .resid)) +
  geom_point()

p1 + p2 + p3
```



LINE: Linearity, Independence, Normal Residuals, Equal Variance

Linearity: We can check the original scatterplot and see if there is a linear trend among the points

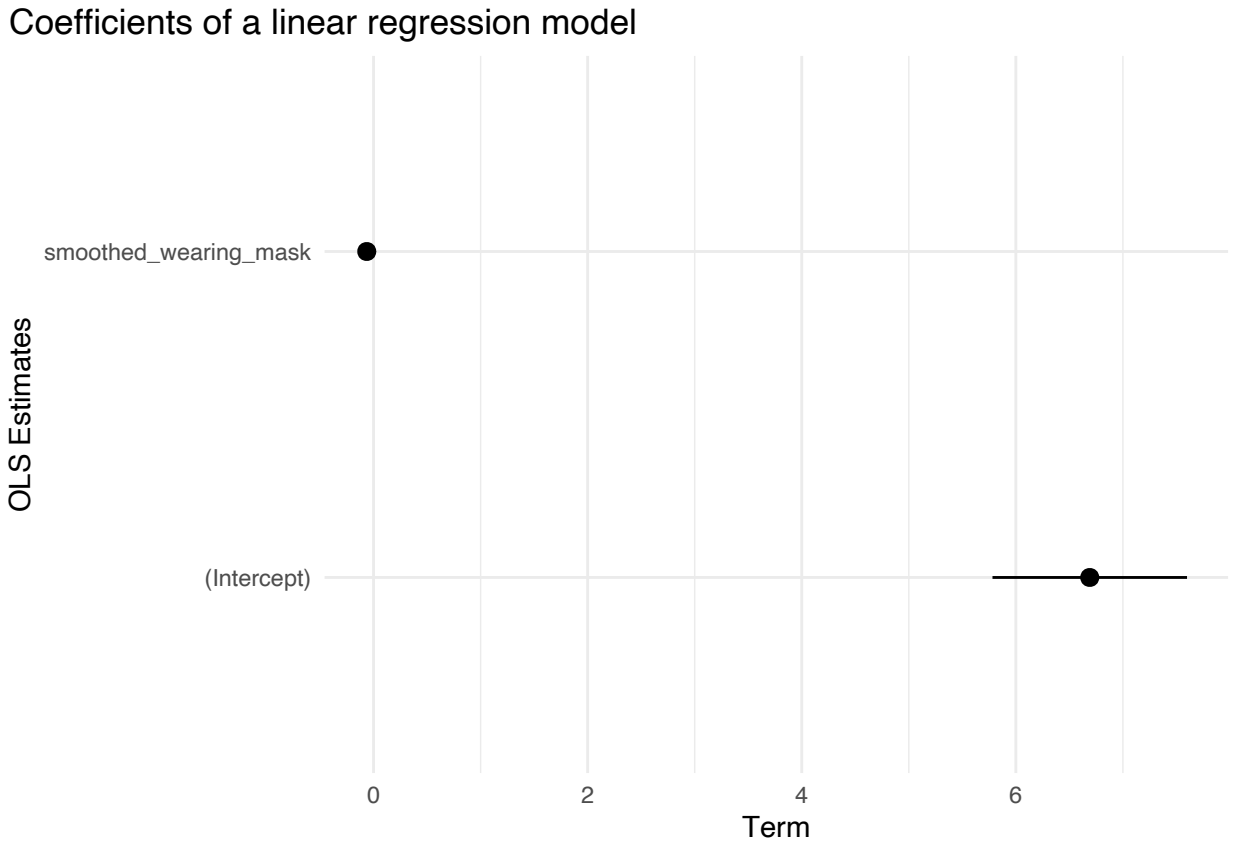
Independence: We can check by plotting an indicator variable against the residuals and there shouldn't be any noticeable trend for it to meet the assumption. You can also think about the context of the data.

Normal Residuals: We can look at the QQ plot of the residuals and if it follows a Normal Distribution, then we are good to go

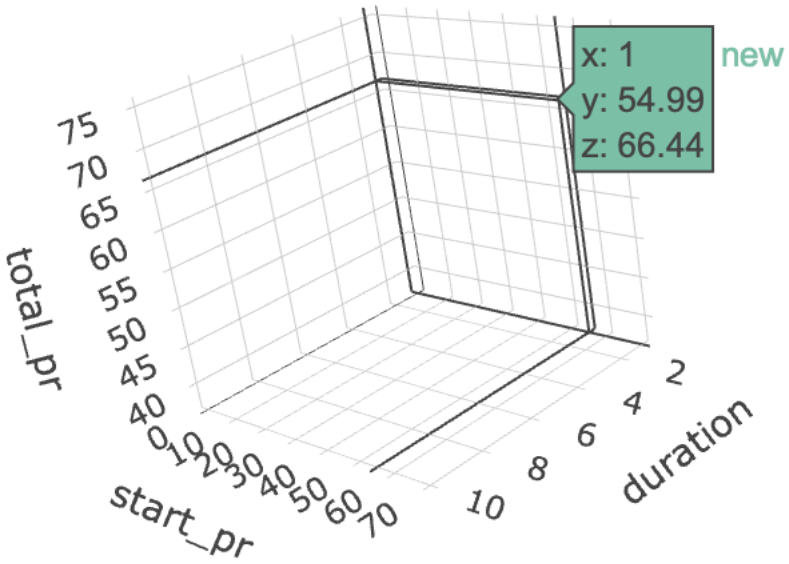
Equal Variance: We can check by looking at the residuals plot and if there isn't a trend among the points, then the assumption is met

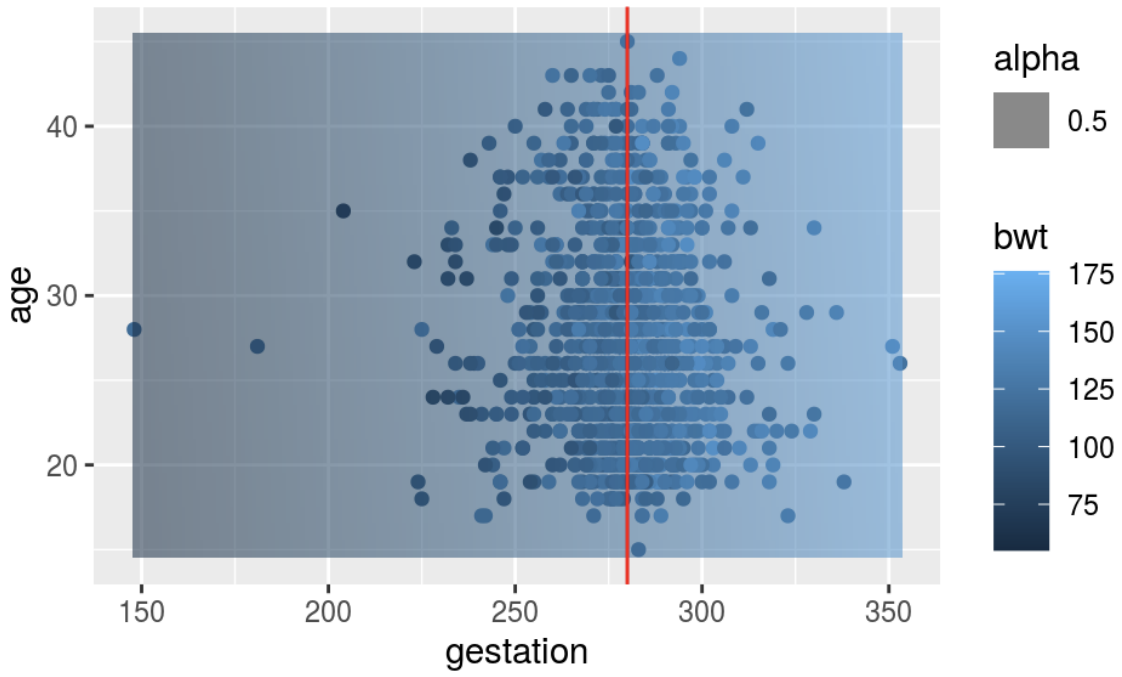
```
tidy_data <- tidy(lm_mod, conf.int = TRUE)

ggplot(data = tidy_data, aes(term,estimate)) +
  geom_pointrange(aes(ymin = conf.low, ymax = conf.high))+
  labs(title = "Coefficients of a linear regression model",x="OLS Estimates",y="Term")+
  coord_flip()+
  theme_minimal()+
  theme(plot.title.position = "plot")
```



Part 2: Multiple Regression Tutorial




age
40
30
20
150
200
250
300
350
gestation
alpha
0.5
bwt
175
150
125
100
75