episode 3

```r
library(tidyverse)
library(tidymodels)
theme_set(theme_minimal(base_size = 14))
dir <- '20210316'
```

```r
library(memer)
m <-
  meme_get('what-is-grief') %>%
  meme_text_top('What is sliced') %>%
  meme_text_bottom('if not contestants persevering')
magick::image_write(m, here::here(dir, 'whatisgrief.png'))
```


```r
import_data <- function(file) {
  here::here(dir, file) %>%
    read_csv() %>%
    mutate(
      avg_peak_frac = str_remove(.data$avg_peak_perc, '[%]') %>% as.numeric(),
      avg_peak_frac = avg_peak_frac * 0.01
    ) %>%
    select(-avg_peak_perc) %>%
    group_by(gamename) %>%
    arrange(yearmonth, .by_group = TRUE) %>%
    mutate(across(
      c(avg, peak, avg_peak_frac),
      list(
        lag1 = dplyr::lag,
        lag2 = ~ dplyr::lag(.x, 2),
        lag12 = ~ dplyr::lag(.x, 12)
      )
    )) %>%
    ungroup()
}

df_trn <- 'sliced_data.csv' %>% import_data()
df_trn <- df_trn %>% mutate(across(volatile, factor))
df_tst <- 'sliced_holdout_data.csv' %>% import_data()
```

no NAs!

```
df_trn %>% skimr::skim()
```

Data summary

Name	Piped data
Number of rows	82373
Number of columns	19
Column type frequency:	
character	2
Date	1
factor	1
numeric	15
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
gamename	0	1	3	81	0	1258	0
month	0	1	3	9	0	12	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
yearmonth	0	1	2012-08-01	2021-02-01	2018-02-01	103

Variable type: factor

skim_variable	n_missing	complete_rate	ordered	n_unique	top_counts
volatile	0	1	FALSE	3	0: 52090, 1: 15505, -1: 14778

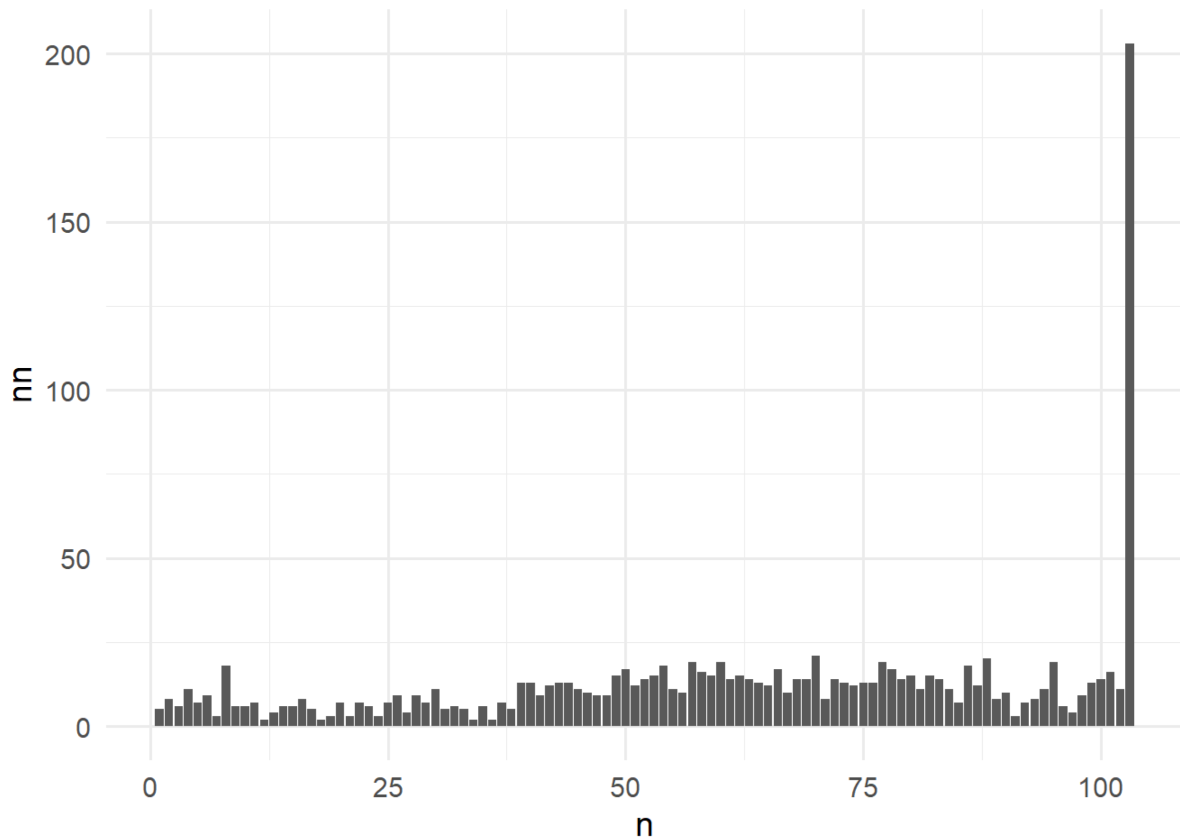
Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
year	0	1.00	2017.37	2.22	2012	2016.00	2018.00	2019.00	2021.00	
avg	0	1.00	2745.77	26619.55	0	53.58	202.51	754.54	1584886.77	
gain	0	1.00	-10.29	3790.65	-250249	-38.18	-1.62	22.24	426446.12	
peak	0	1.00	5411.54	50360.09	0	138.00	498.00	1703.00	3236027.00	
month_num	0	1.00	6.54	3.52	1	3.00	7.00	10.00	12.00	
avg_peak_frac	106	1.00	0.43	0.13	0	0.35	0.44	0.51	0.89	
avg_lag1	1258	0.98	2735.63	26632.41	0	53.61	201.77	748.95	1584886.77	
avg_lag2	2511	0.97	2723.25	26637.46	0	53.54	200.72	741.34	1584886.77	
avg_lag12	14592	0.82	2653.47	26875.40	0	53.57	193.42	699.86	1584886.77	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
peak_lag1	1258	0.98	5398.27	50475.05	0	138.00	496.00	1696.00	3236027.00	
peak_lag2	2511	0.97	5382.87	50591.72	0	138.00	494.00	1685.00	3236027.00	
peak_lag12	14592	0.82	5295.10	51872.78	0	140.00	483.00	1616.00	3236027.00	
avg_peak_frac_lag1	1364	0.98	0.43	0.13	0	0.35	0.44	0.51	0.89	
avg_peak_frac_lag2	2617	0.97	0.42	0.13	0	0.35	0.44	0.51	0.89	
avg_peak_frac_lag12	14698	0.82	0.42	0.13	0	0.34	0.43	0.51	0.89	

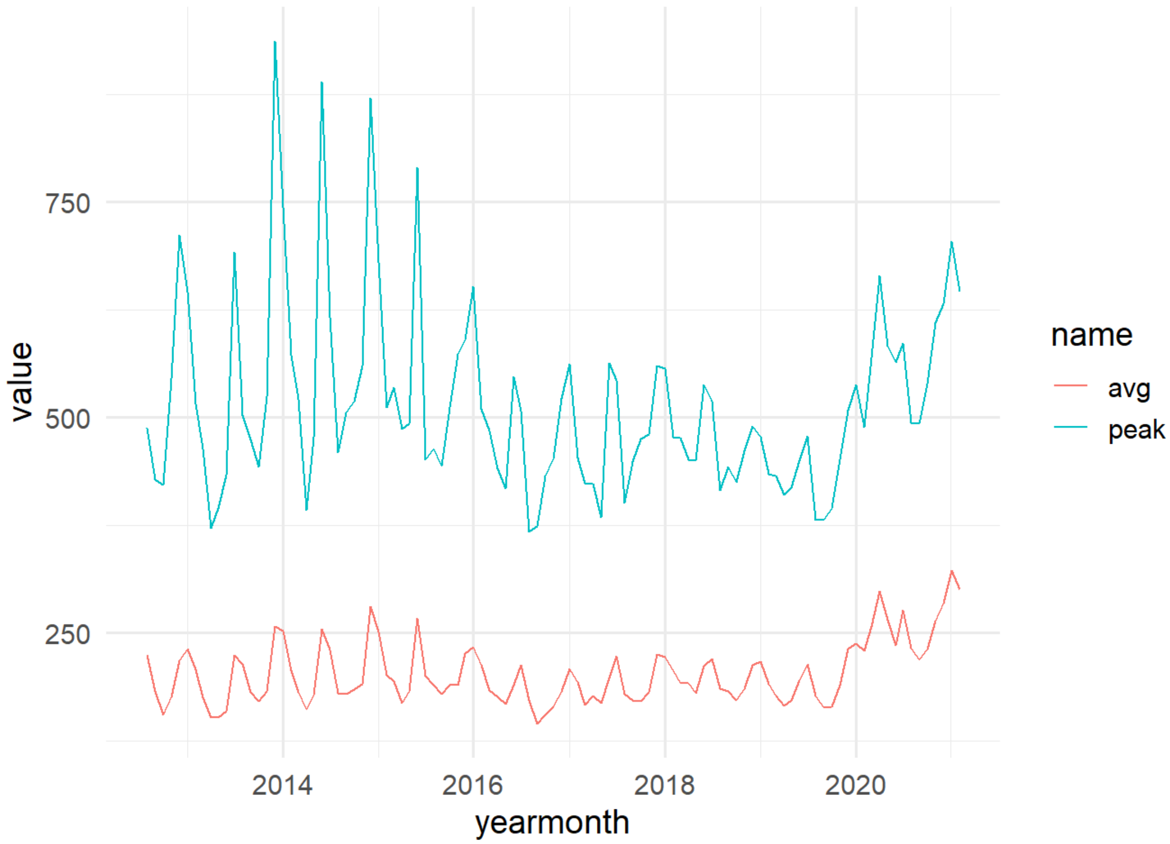
lots of games in all 103 months

```
df_trn %>%
  count(gamename, sort = TRUE) %>%
  count(n, name = 'nn') %>%
  ggplot() +
  aes(n, nn) +
  geom_col()
```



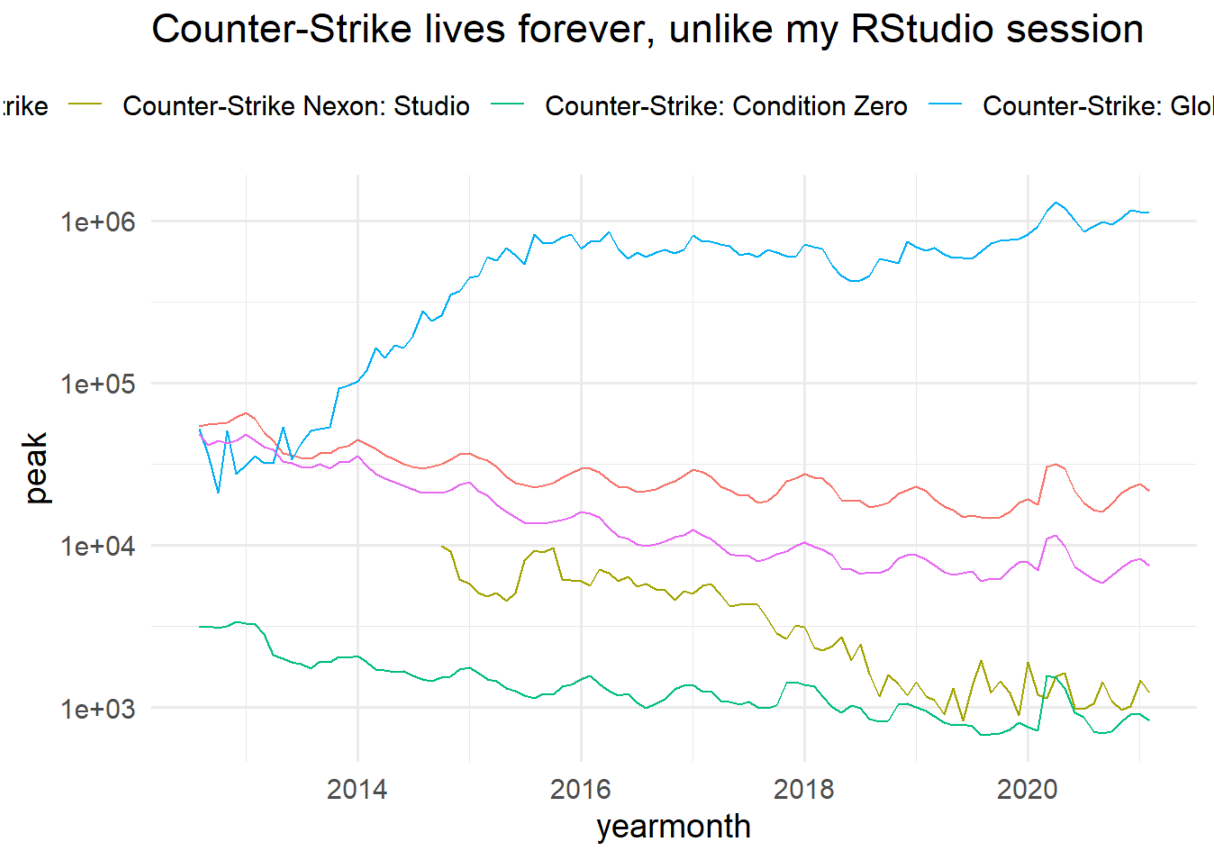
median looks better than mean

```
df_trn %>%
  group_by(yearmonth) %>%
  summarize(
    across(c(avg, peak), median, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  pivot_longer(-yearmonth) %>%
  ggplot() +
  aes(x = yearmonth, y = value, color = name) +
  geom_line()
```



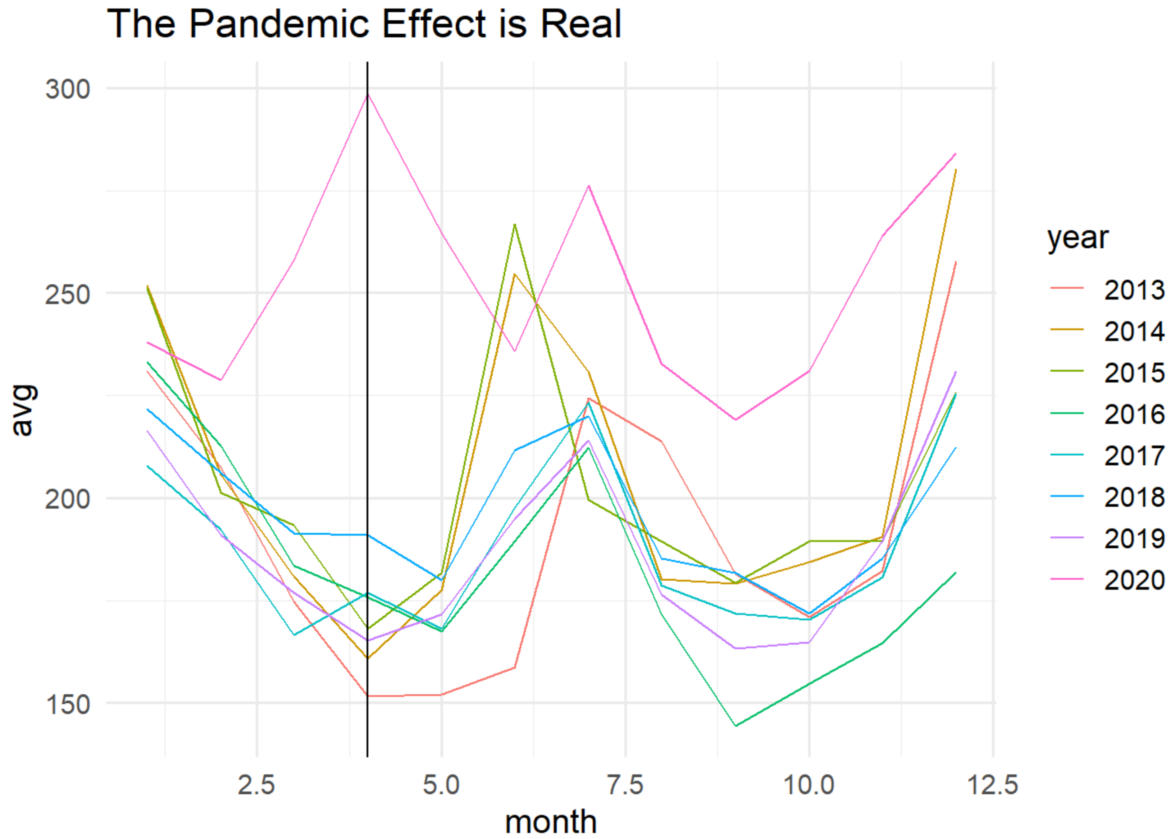
counterstrike viz (for a golden feature?)

```
df_trn %>%
  filter(gamename %>% str_detect('Counter')) %>%
  # count(gamename)
  ggplot() +
  aes(x = yearmonth, y = peak, color = gamename, group = gamename) +
  geom_line() +
  scale_y_log10() +
  theme(
    legend.position = 'top'
  ) +
  labs(
    title = 'Counter-Strike lives forever, unlike my RStudio session'
  )
```

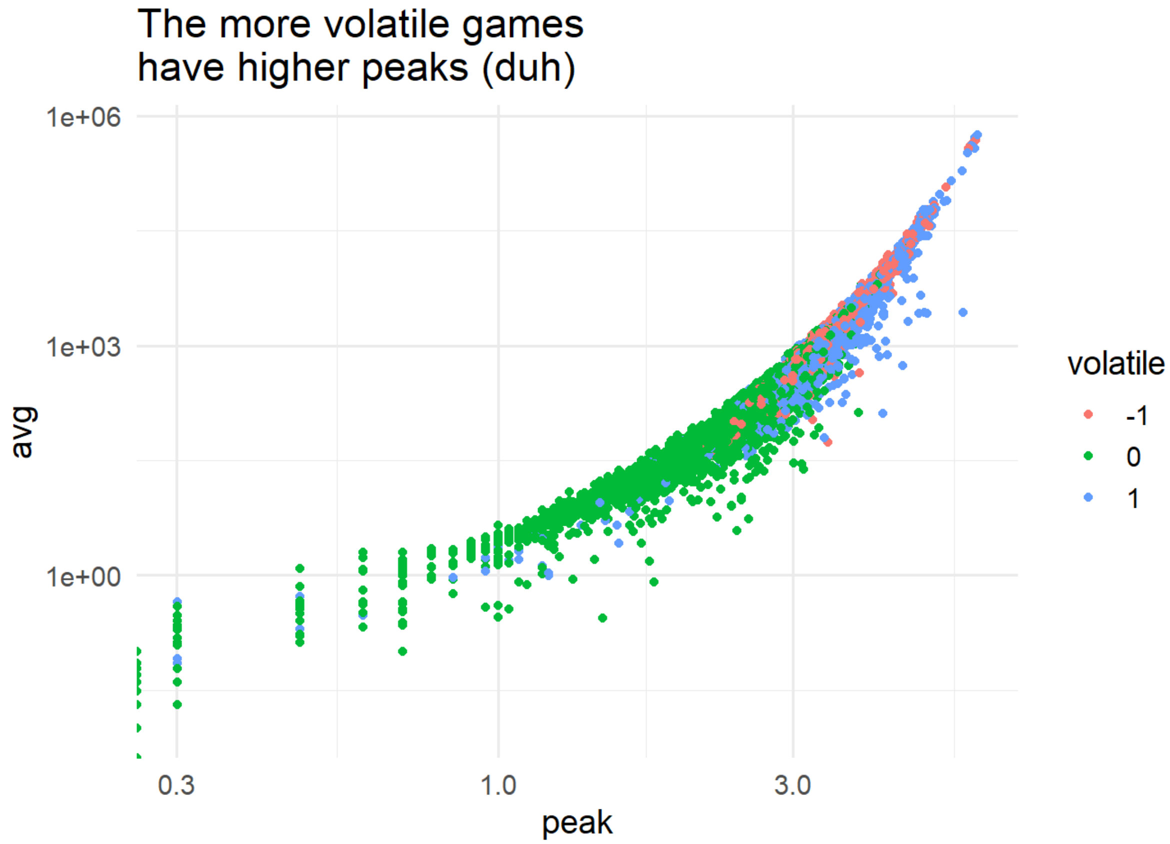


march 2020 clearly is a large outlier, but that doesn't actually mean more volatility.

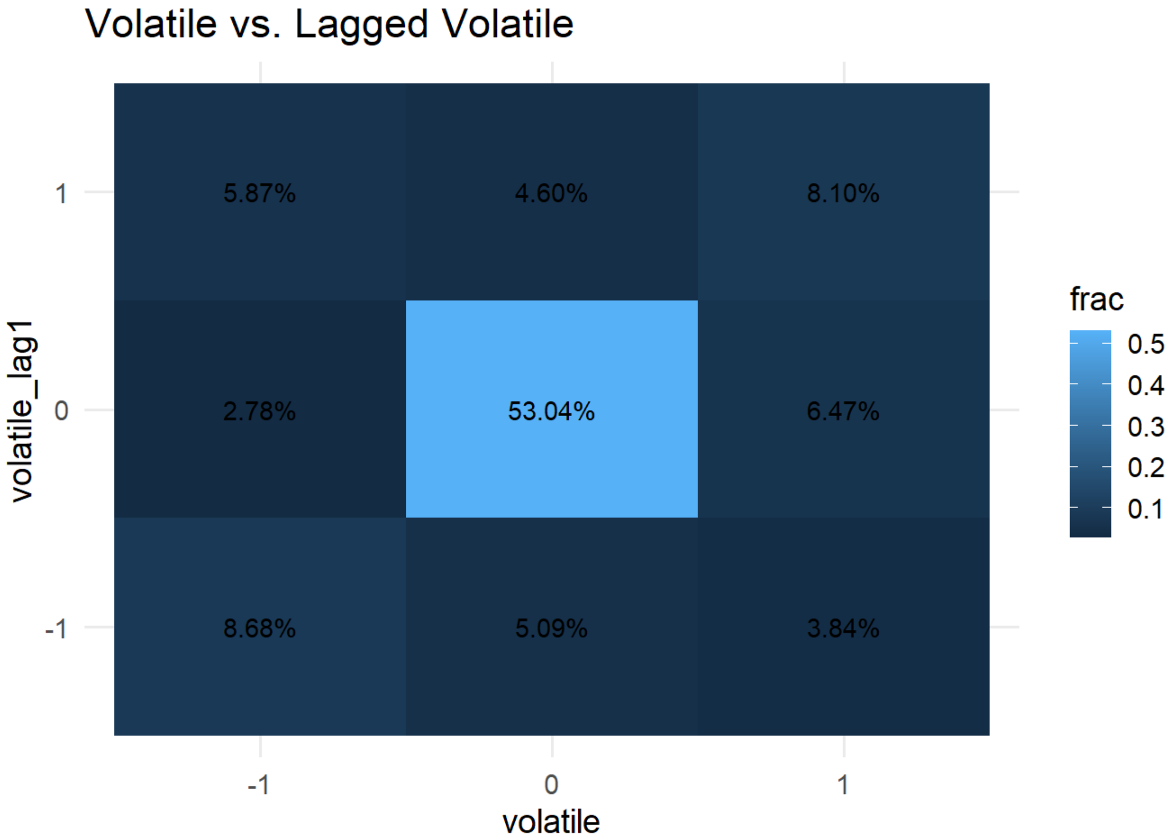
```
# df_trn %>% count(month)
# df_trn %>% count(year)
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  group_by(yearmonth, year, month) %>%
  summarize(
    across(c(avg, peak), median, na.rm = TRUE)
  ) %>%
  ungroup() %>%
  mutate(
    across(year, factor),
    month = ordered(month, levels = month.name) %>% as.integer()
  ) %>%
  ggplot() +
  aes(x = month, y = avg, color = year, group = year) +
  geom_line() +
  geom_vline(aes(xintercept = 4)) +
  labs(title = 'The Pandemic Effect is Real')
```



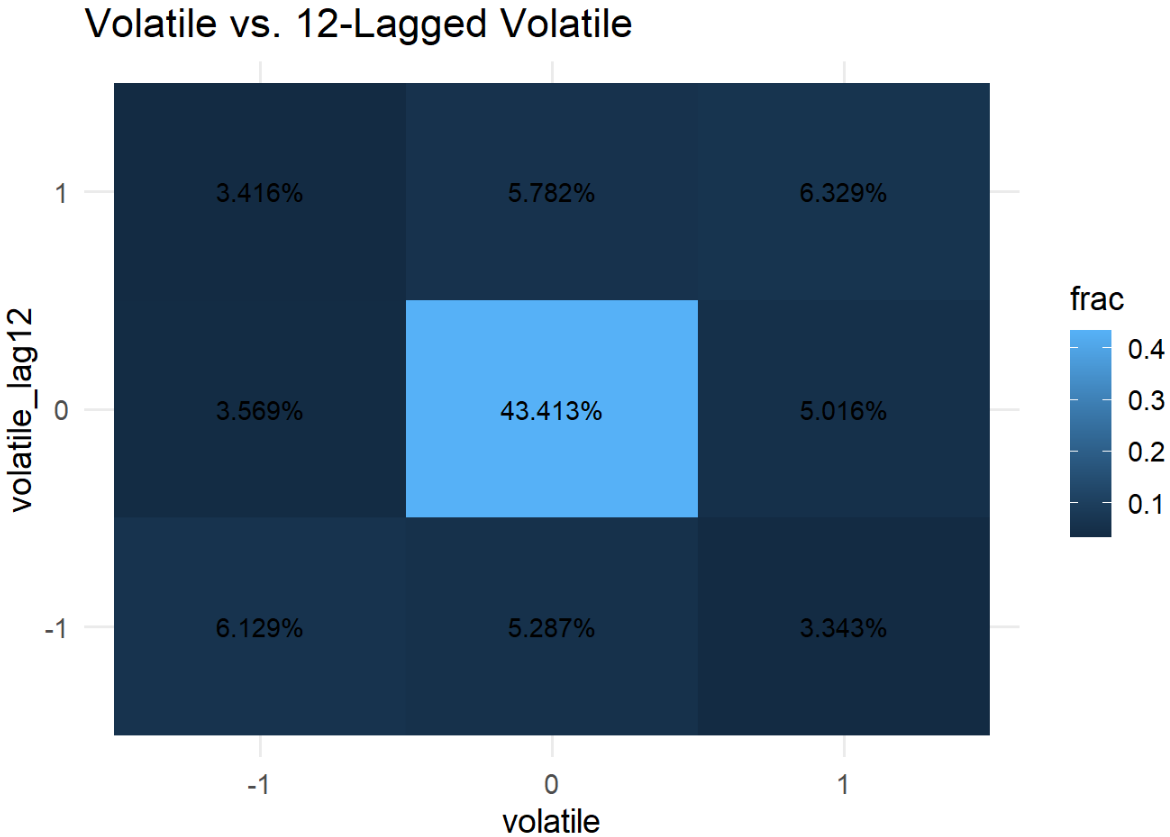

```
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  sample_frac(0.1) %>%
  mutate(across(peak, log10)) %>%
  # filter(is.na(peak))
  ggplot() +
  aes(x = peak, y = avg) +
  # ggridges::geom_density_ridges()
  scale_x_log10() +
  scale_y_log10() +
  geom_point(aes(color = volatile)) +
  labs(
    title = 'The more volatile games\nhave higher peaks (duh)'
  )
```



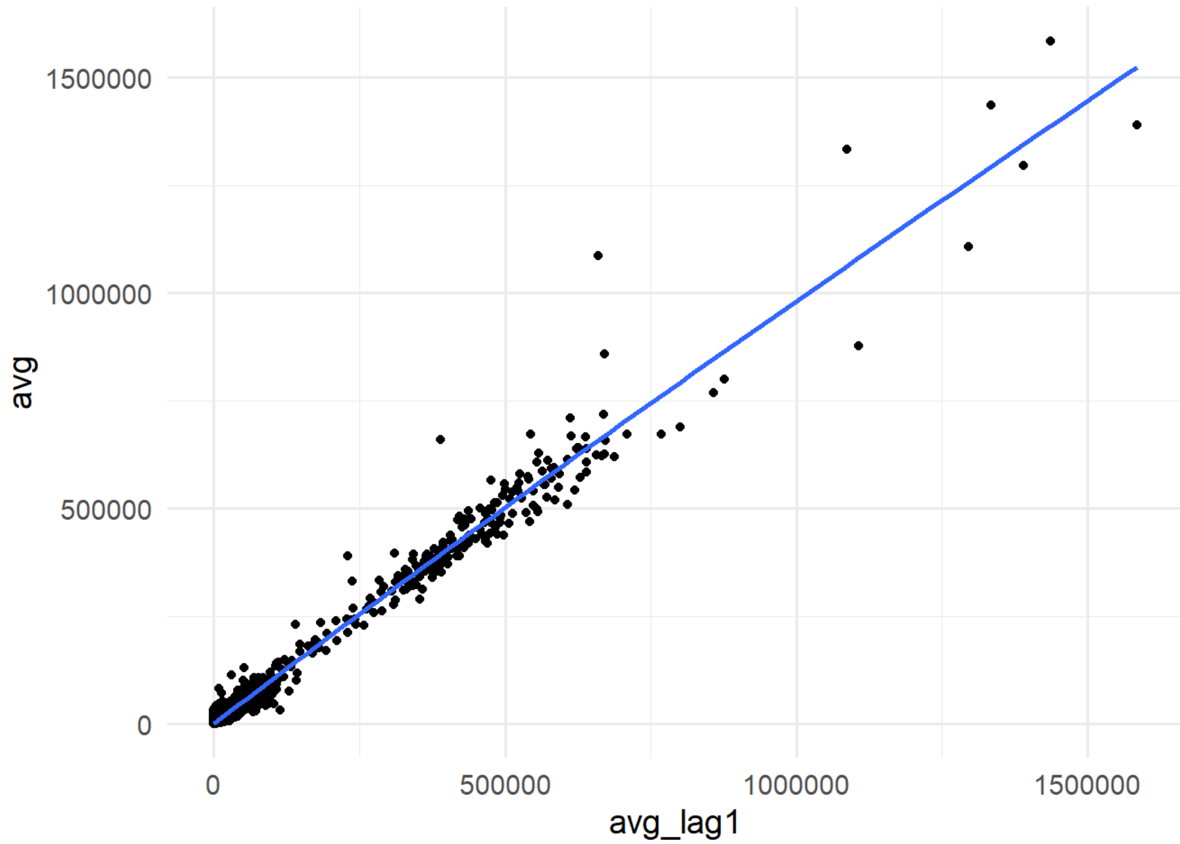

```
df_trn %>%
  group_by(gamename) %>%
  arrange(yearmonth, .by_group = TRUE) %>%
  mutate(across(volatile, list(lag1 = dplyr::lag, lag13 = ~dplyr::lag(.x, 13)))) %>%
  ungroup() %>%
  count(volatile, volatile_lag1) %>%
  mutate(frac = n / sum(n)) %>%
  drop_na() %>%
  ggplot() +
  aes(x = volatile, y = volatile_lag1) +
  geom_tile(aes(fill = frac)) +
  geom_text(aes(label = scales::percent(frac))) +
  labs(
    title = 'Volatile vs. Lagged Volatile'
  )
```



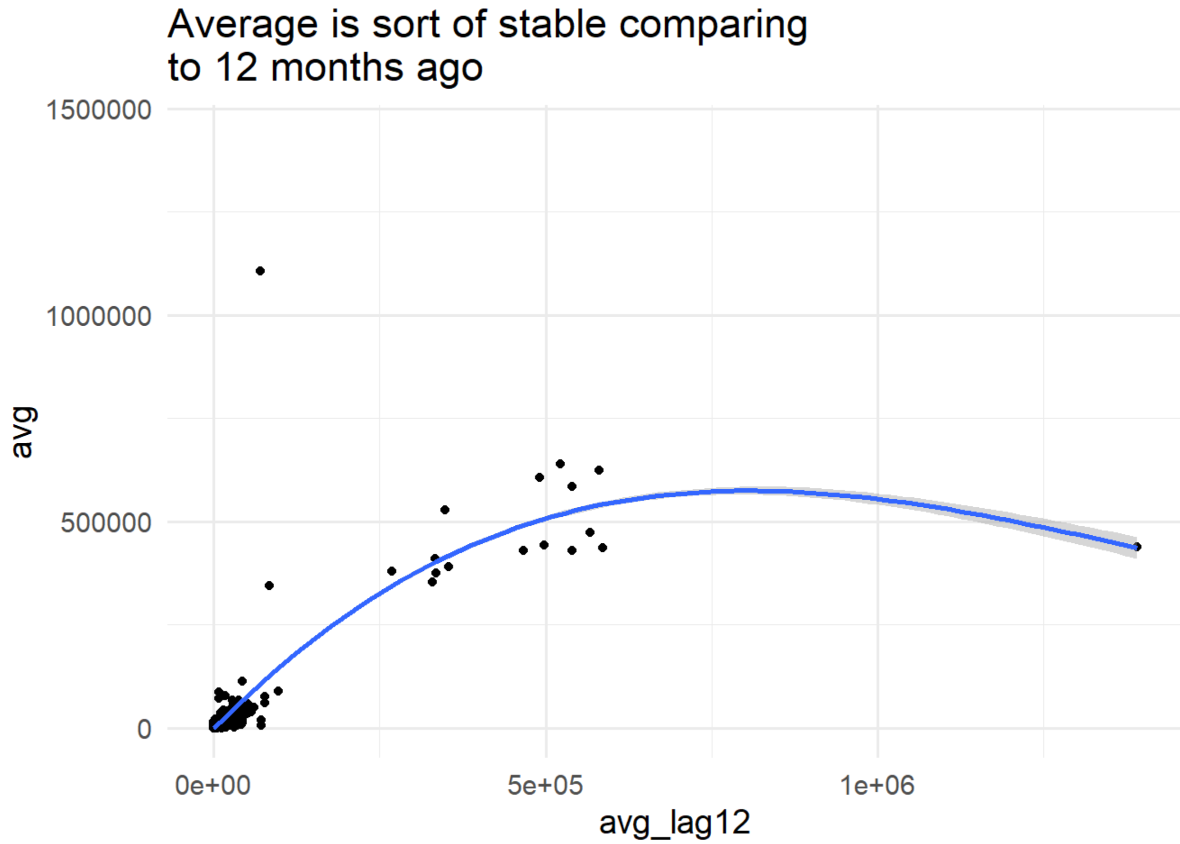

```
df_trn %>%
  group_by(gamename) %>%
  arrange(yearmonth, .by_group = TRUE) %>%
  mutate(across(volatile, list(lag1 = dplyr::lag, lag12 = ~dplyr::lag(.x, 12)))) %>%
  ungroup() %>%
  count(volatile, volatile_lag12) %>%
  mutate(frac = n / sum(n)) %>%
  drop_na() %>%
  ggplot() +
  aes(x = volatile, y = volatile_lag12) +
  geom_tile(aes(fill = frac)) +
  geom_text(aes(label = scales::percent(frac))) +
  labs(
    title = 'Volatile vs. 12-Lagged Volatile'
  )
```



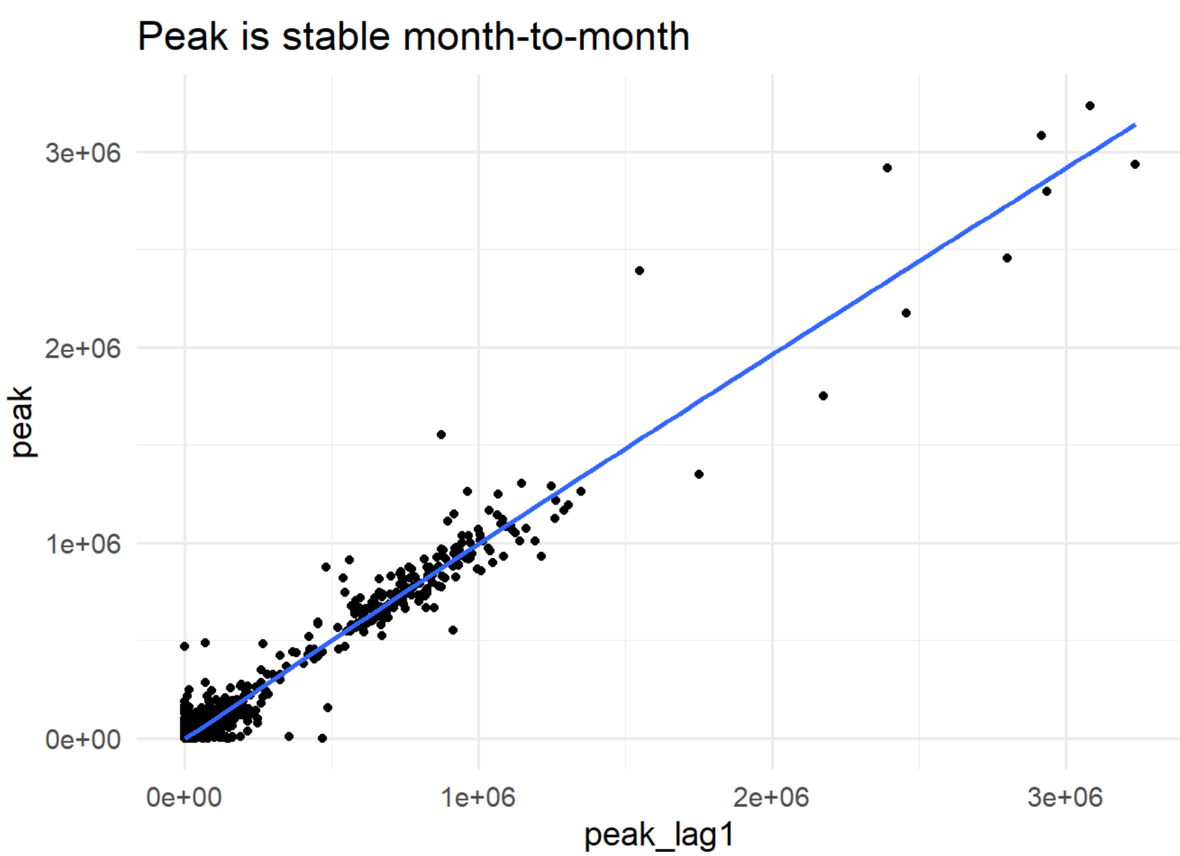

```
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  # # sample_frac(0.1) %>%
  # select(gamename, yearmonth, matches('(avg|peak)'), matches('(avg|peak)_lag1')) %>%
  # pivot_longer(
  #   -c(gamename, yearmonth)
  # ) %>%
  ggplot() +
  aes(x = avg_lag1, y = avg) +
  geom_point() +
  geom_smooth()
```



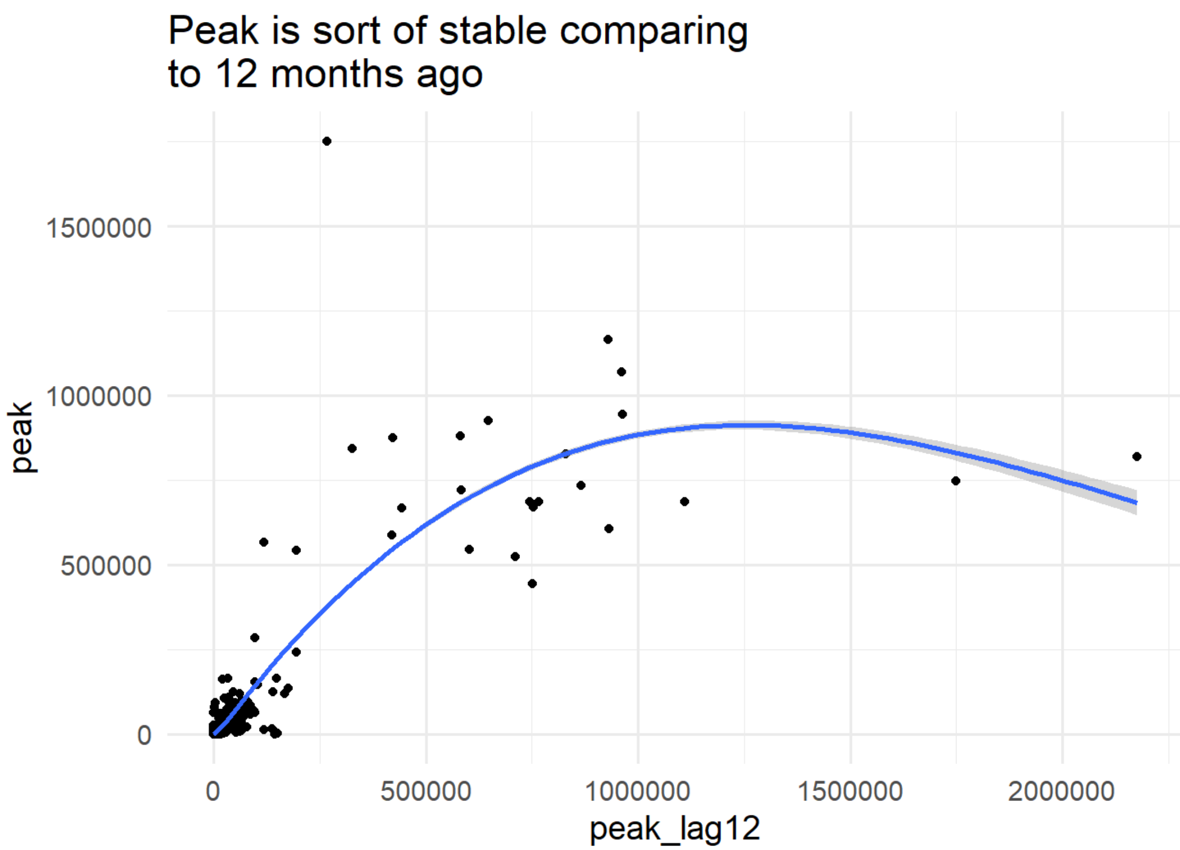

```
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  sample_frac(0.1) %>%
  ggplot() +
  aes(x = avg_lag12, y = avg) +
  geom_point() +
  geom_smooth() +
  labs(
    title = 'Average is sort of stable comparing\nto 12 months ago'
  )
```



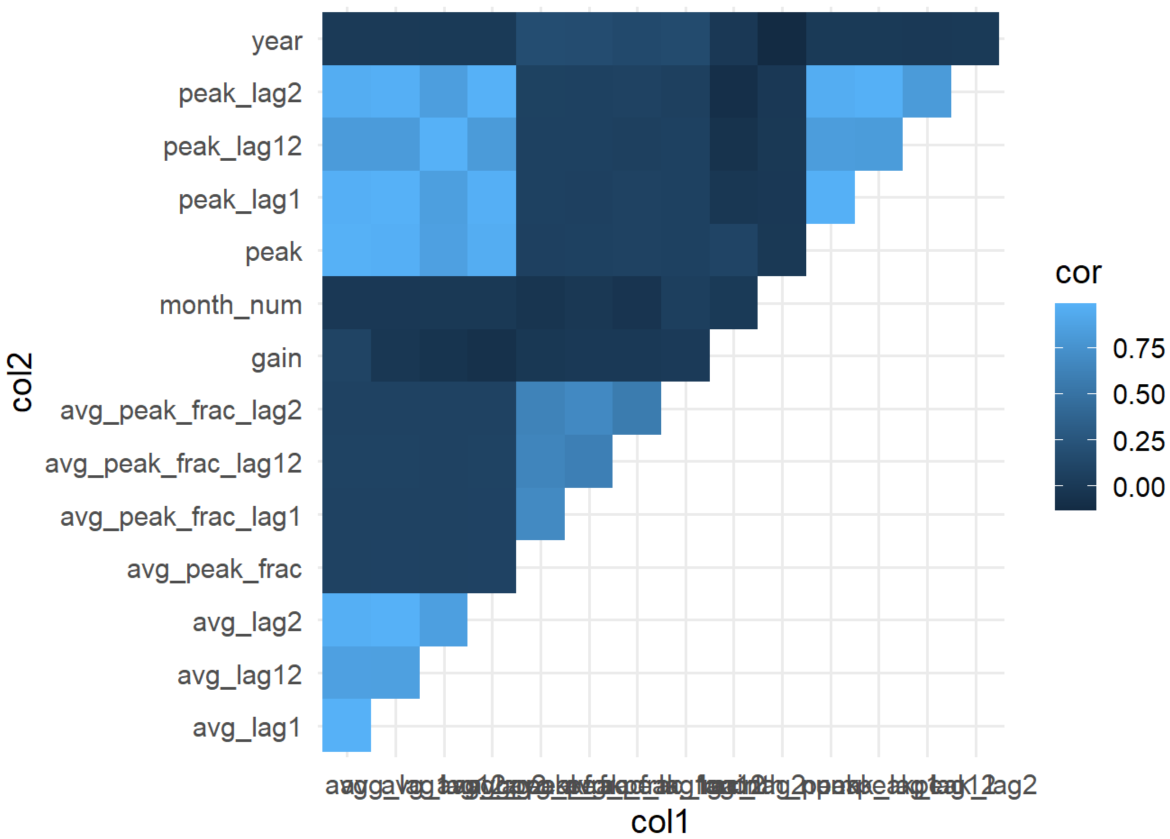

```
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  ggplot() +
  aes(x = peak_lag1, y = peak) +
  geom_point() +
  geom_smooth() +
  labs(
    title = 'Peak is stable month-to-month'
  )
```



```
df_trn %>%
  filter(year > 2012, year < 2021) %>%
  sample_frac(0.1) %>%
  ggplot() +
  aes(x = peak_lag12, y = peak) +
  geom_point() +
  geom_smooth() +
  labs(
    title = 'Peak is sort of stable comparing\nto 12 months ago'
  )
```



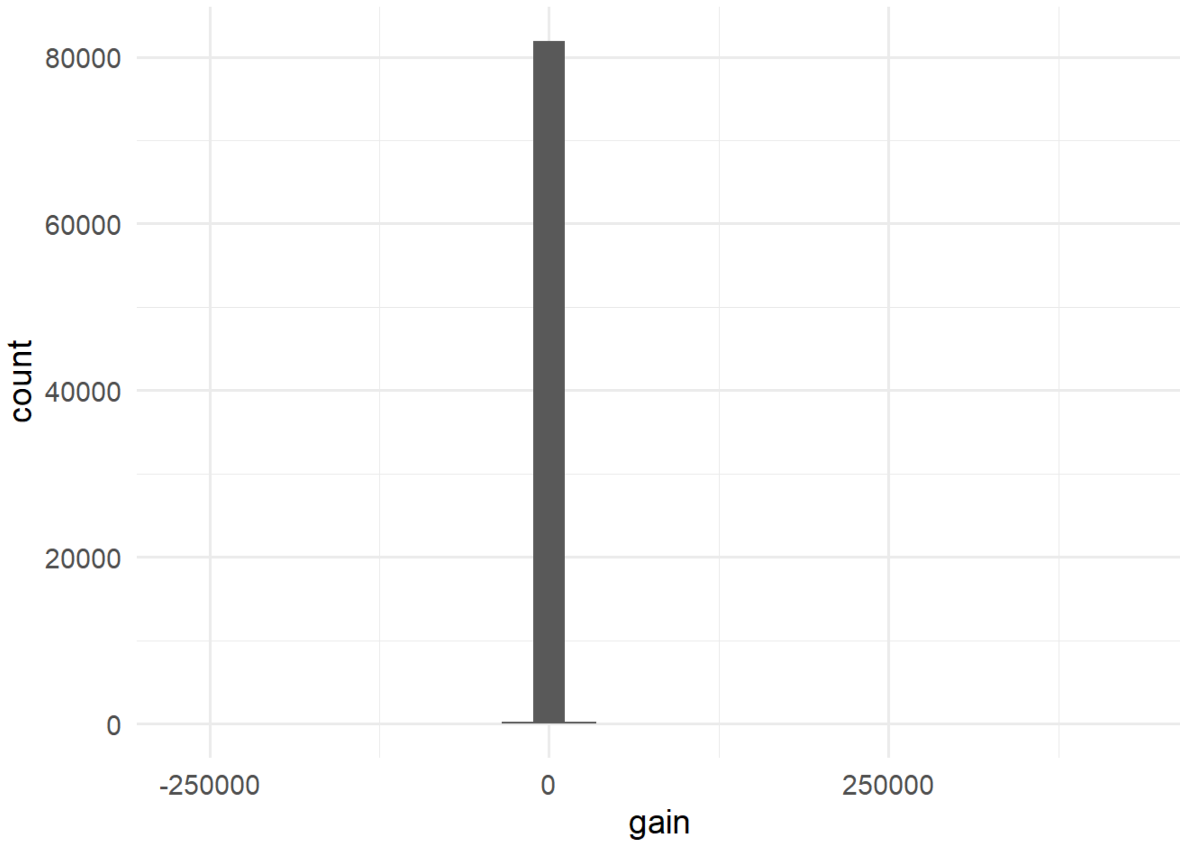

```
df_trn %>%
  select(where(is.numeric)) %>%
  corrr::correlate() %>%
  rename(col1 = rowname) %>%
  pivot_longer(
    -col1,
    names_to = 'col2',
    values_to = 'cor'
  ) %>%
  filter(col1 < col2) %>%
  # filter(cor > 0.5) %>%
  ggplot() +
  aes(x = col1, y = col2) +
  geom_tile(aes(fill = cor))
```




```
df_trn %>% select(gain)
```

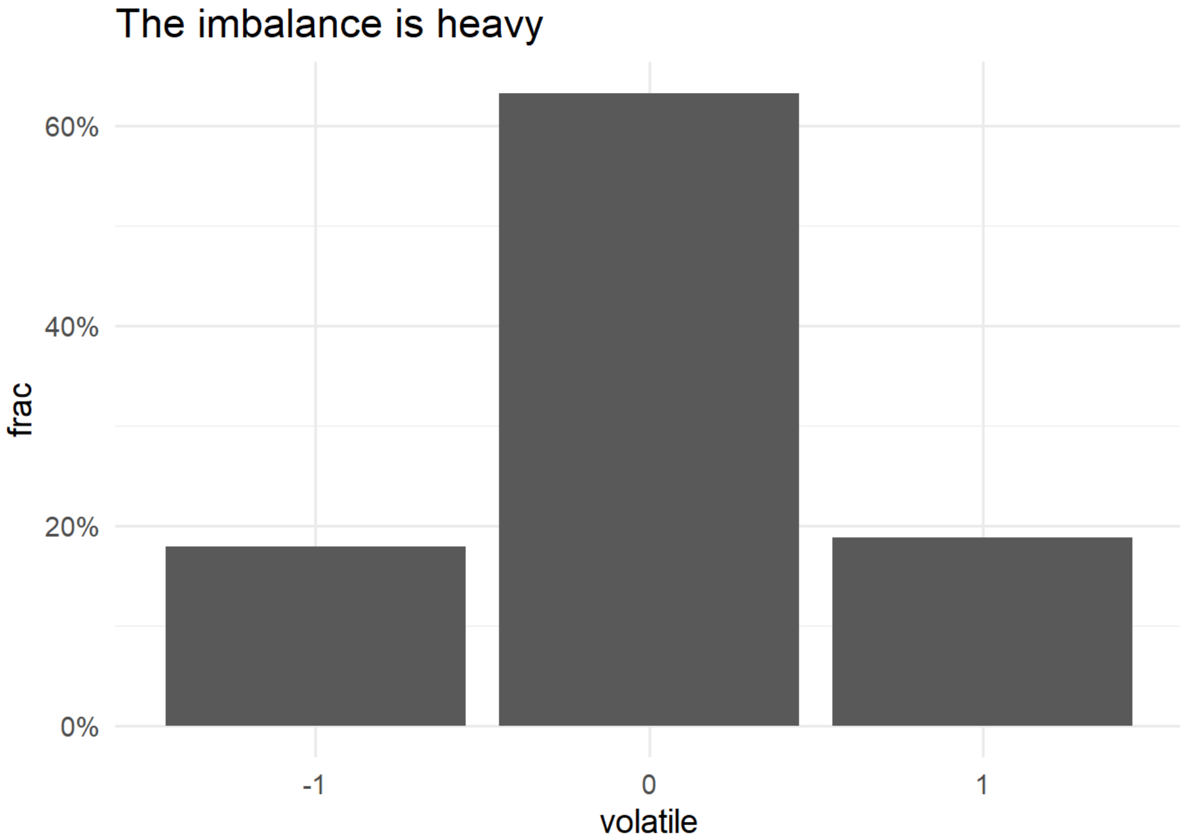
```
## # A tibble: 82,373 x 1
##       gain
##      <dbl>
##  1 -4727.
##  2 -2022.
##  3  -142.
##  4   -69.6
##  5   -55.9
##  6   -13.8
##  7   -11.9
##  8   -12.2
##  9   124.
## 10  2392.
## # ... with 82,363 more rows
```

```
df_trn %>%
  ggplot() +
  aes(x = gain) +
  geom_histogram()
```



imbalance stuffs

```
df_trn %>%
  count(volatile) %>%
  mutate(frac = n / sum(n)) %>%
  ggplot() +
  aes(x = volatile, y = frac) +
  geom_col() +
  scale_y_continuous(labels = scales::percent) +
  labs(
    title = 'The imbalance is heavy'
  )
```



The 3 most volatile months were early on (pre-2015). april 2020 is 88 / 103

```
df_trn %>%
  count(yearmonth, is_zero = ifelse(volatile == 0, TRUE, FALSE)) %>%
  group_by(yearmonth) %>%
  mutate(frac = n / sum(n)) %>%
  ungroup() %>%
  filter(!is_zero) %>%
  arrange(-frac) %>%
  mutate(rnk = row_number(-frac)) %>%
  filter(yearmonth == '2020-03-01')
```

```
## # A tibble: 1 x 5
##   yearmonth  is_zero     n  frac   rnk
##   <date>     <lgl>   <int> <dbl> <int>
## 1 2020-03-01 FALSE     413 0.353    76
```

```
df_trn %>%
  mutate(
    is_pandemic = ifelse(yearmonth == '2020-04-01', TRUE, FALSE)
  ) %>%
  filter(is_pandemic) %>%
  count(volatile) %>%
  mutate(frac = n / sum(n))
```

```
## # A tibble: 3 x 3
##   volatile     n  frac
## * <fct>    <int> <dbl>
## 1 -1         145 0.123
## 2 0          785 0.666
## 3 1          248 0.211
```

```
rec <-
  recipe(volatile ~ ., data = df_trn) %>%
  step_rm(year, month) %>%
  step_date(yearmonth, features = c('month', 'year')) %>%
  step_rm(yearmonth) %>%
  update_role(gamename, new_role = 'id') %>%
  # step_impute_knn(all_predictors()) # %>%
  step_impute_mean(all_numeric_predictors())
  # themis::step_smote(volatile)



jui <- rec %>% prep() %>% juice()

rec_dummy <-
  recipe(volatile ~ avg + peak, data = df_trn)
jui_dummy <- rec %>% prep() %>% juice()

# jui %>% skimr::skim()
# rec %>% prep() %>% bake(df_tst)
```

```
# set.seed(6*6*6)
# Avoid the data leakage!
# folds <- df_trn %>% group_vfold_cv(group = 'gamename')

# Does random forest work for multinomial?
spec_glmnet <-
  multinom_reg(mixture = 0.5, penalty = 0.001) %>%
  set_mode('classification') %>%
  set_engine('glmnet')
spec_glmnet

wf_glmnet <-
  workflow() %>%
  add_recipe(rec_dummy) %>%
  add_model(spec_glmnet)
wf_glmnet

# why this happen?!?
fit_glmnet <- wf_glmnet %>% fit(df_trn)
fit_glmnet

preds_trn_glmnet <- fit_glmnet %>% predict(df_trn)
```

```
# reading up on nnet docs...
spec <-
  multinom_reg() %>%
  set_mode('classification') %>%
  # RIP me no keras
  set_engine('nnet')

wf <-
  workflow() %>%
  add_recipe(rec) %>%
  add_model(spec)

fit <- wf %>% fit(df_trn)
preds_trn <- fit %>% predict(df_trn) %>% bind_cols(df_trn %>% select(volatile))
# 0.817 accuracy ok buddy
preds_trn %>% accuracy(volatile, .pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.826
```

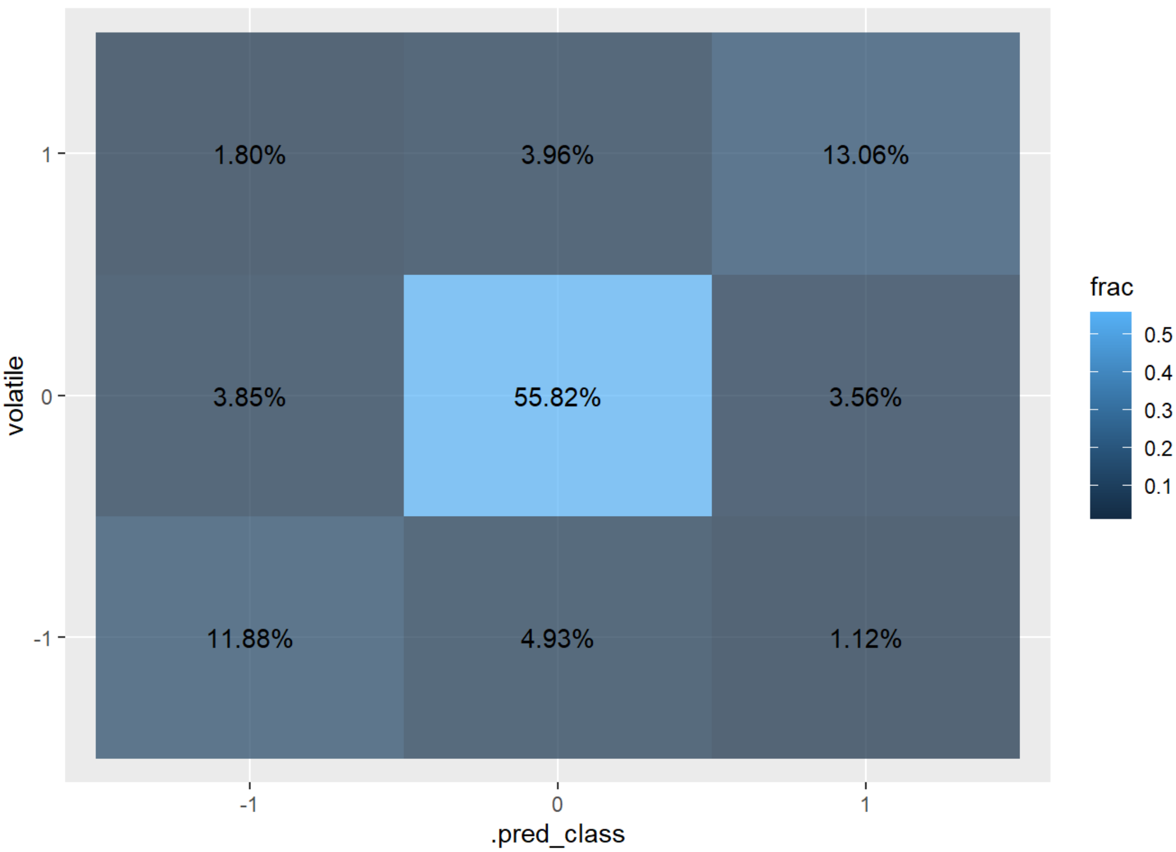
```
spec_nn <-
  mlp() %>%
  set_mode('classification') %>%
  # RIP me no keras
  set_engine('nnet')

wf_nn <-
  workflow() %>%
  add_recipe(rec) %>%
  add_model(spec_nn)

fit_nn <- wf_nn %>% fit(df_trn)

preds_trn_nn <- fit_nn %>% predict(df_trn) %>% bind_cols(df_trn %>% select(volatile))
```

```
preds_trn_nn %>%
  count(.pred_class, volatile) %>%
  mutate(frac = n / sum(n)) %>%
  ggplot() +
  aes(.pred_class, volatile) +
  geom_tile(aes(fill = frac), alpha = 0.7) +
  geom_text(aes(label = scales::percent(frac)))
```




```
# 0.830 accuracy cool story bro
preds_trn_nn %>% accuracy(volatile, .pred_class)
```

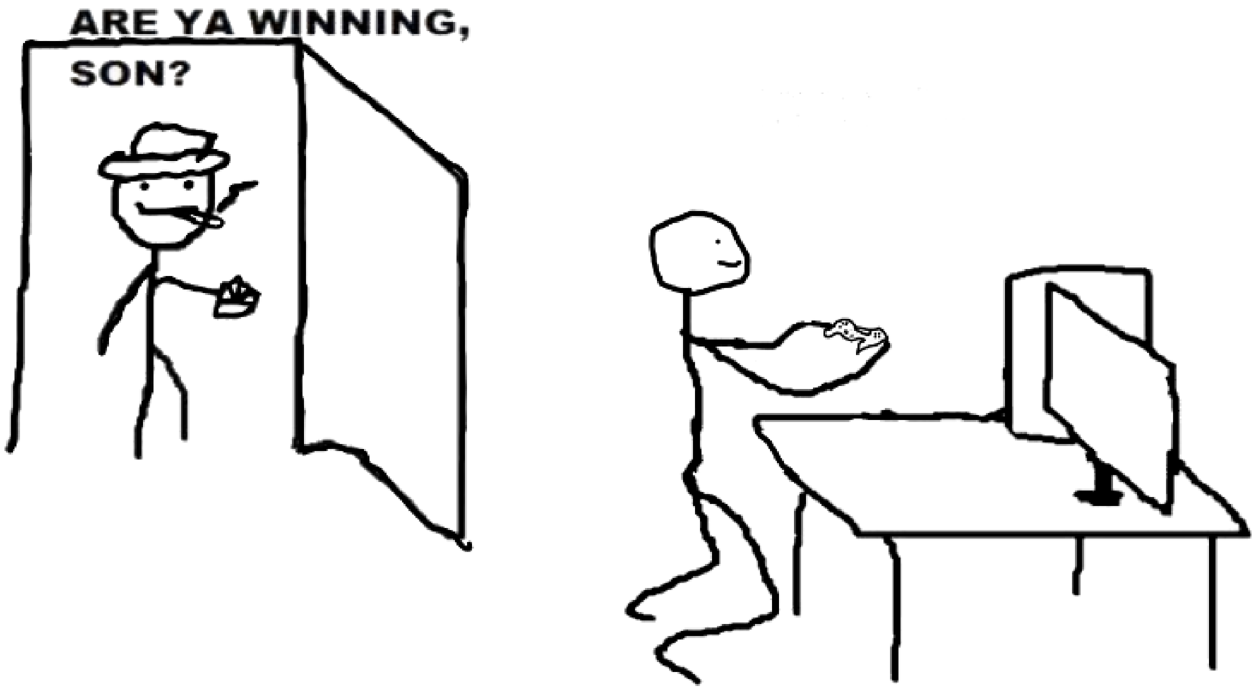
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.808
```

```
preds_nn <- fit_nn %>% predict(df_tst)
preds_nn
```

```
## # A tibble: 103 x 1
##    .pred_class
##    <fct>
##  1 1
##  2 -1
##  3 1
##  4 1
##  5 1
##  6 1
##  7 1
##  8 -1
##  9 -1
## 10 -1
## # ... with 93 more rows
```

```
write_csv(preds_nn, 'holdout_preds.csv')
```

```
# didn't get to this
# params_grid <-
#   grid_latin_hypercube(
#     # parameters(spec),
#     finalize(mtry(), jui),
#     size = 10
#   )
#
# res_tune <-
#   tune_grid(
#     wf_rf,
#     resamples = folds,
#     metrics = yardstick::accuracy
#     control = control_grid(verbose = TRUE)
#   )
#
# params_best <- res_tune %>% select_best('accuracy')
# wf_best <- wf_rf %>% finalize_workflow(params_best)
# fit_best <- wf_best %>% fit(df_trn)
```


ARE YA WINNING,
SON?