

Tidymodels

```
library(tidymodels)

## Registered S3 method overwritten by 'tune':
##   method           from
##   required_pkgs.model_spec parsnip

## — Attaching packages —————— tidymodels 0.1.4 —

## ✓ broom      0.7.10    ✓ rsample     0.1.0
## ✓ dials      0.0.10    ✓ tune        0.1.6
## ✓ infer       1.0.0     ✓ workflows   0.2.4
## ✓ modeldata   0.1.1     ✓ workflowsets 0.1.0
## ✓ parsnip     0.1.7     ✓ yardstick   0.0.8
## ✓ recipes     0.1.17

## — Conflicts —————— tidymodels_conflicts() —

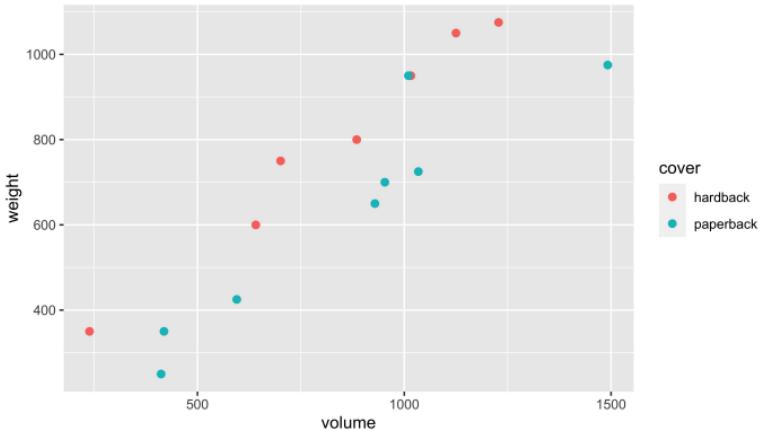
## x scales::discard()  masks purrr::discard()
## x dplyr::filter()    masks stats::filter()
## x recipes::fixed()   masks stringr::fixed()
## x dplyr::lag()       masks stats::lag()
## x rsample::populate() masks Rcpp::populate()
## x yardstick::spec()  masks readr::spec()
## x recipes::step()    masks stats::step()
## • Dig deeper into tidy modeling with R at https://www.tmwr.org
```

Book data

```
(books = DAAG::allbacks %>%
  as_tibble() %>%
  select(-area) %>%
  mutate(
    cover = forcats::fct_recode(
      cover,
      "hardback" = "hb",
      "paperback" = "pb"
    )
  )
)
```

```
## # A tibble: 15 × 3
##   volume weight cover
##     <dbl>   <dbl> <fct>
## 1     885     800 hardback
## 2    1016     950 hardback
## 3    1125    1050 hardback
## 4     239     350 hardback
## 5     701     750 hardback
## 6     641     600 hardback
## 7    1228    1075 hardback
## 8     412     250 paperback
```

```
ggplot(books, aes(x=volume, y=weight, color = cov)
  geom_point(size=2)
```



Building a tidymodel

```
linear_reg()
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Building a tidymodel

```
linear_reg() %>%  
  set_engine("lm")  
  
## Linear Regression Model Specification (regression)  
##  
## Computational engine: lm
```

Building a tidymodel

```
linear_reg() %>%
  set_engine("lm") %>%
  fit(weight ~ volume * cover, data = books)

## parsnip model object
##
## Fit time: 10ms
##
## Call:
## stats::lm(formula = weight ~ volume * cover, da
##
## Coefficients:
##             (Intercept)          volume
##             161.58654        0.76159
##   coverpaperback  volume:coverpaperback
##             -120.21407       -0.07573
```

```
lm(weight ~ volume * cover, data = books)

##
## Call:
## lm(formula = weight ~ volume * cover, data = books)
##
## Coefficients:
##             (Intercept)          volume
##             161.58654        0.76159
##   coverpaperback  volume:coverpaperback
##             -120.21407       -0.07573
```

Tidy model objects



```
summary(lm(weight ~ volume * cover, data = books))

##
## Call:
## lm(formula = weight ~ volume * cover, data = books)
##
## Residuals:
##    Min      1Q Median      3Q     Max 
## -89.67 -32.07 -21.82  17.94 215.91 
## 
## Coefficients:
##             Estimate Std. Error t value
## (Intercept) 161.58654   86.51918   1.868
## volume       0.76159    0.09718   7.837
## coverpaperback -120.21407  115.65899  -1.039
## volume:coverpaperback -0.07573   0.12802  -0.592
##             Pr(>|t|)    
## (Intercept) 0.0887 .
## volume      7.94e-06 ***
## coverpaperback 0.3209
## volume:coverpaperback 0.5661
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 80.41 on 11 degrees of freedom
## Multiple R-squared:  0.9297, Adjusted R-squared:  0.9105 
## F-statistic: 48.5 on 3 and 11 DF,  p-value: 1.245e-06
```

```
lm_tm = linear_reg() %>%
  set_engine("lm") %>%
  fit(weight ~ volume * cover, data = books)
```

```
summary(lm_tm)
```

```
## # A tibble: 4 × 5
##   term            estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) 1.62e+2   86.5        1.87    8.87e-2
## 2 volume       7.62e-1   0.0972    7.84    7.94e-6
```

```
broom::tidy(lm_tm)
```

```
## # A tibble: 4 × 5
##   term            estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept) 1.62e+2   86.5        1.87    8.87e-2
## 2 volume       7.62e-1   0.0972    7.84    7.94e-6
```

Tidy model statistics

```
broom::glance(lm(weight ~ volume * cover, data = books))
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic   p.value     df
##       <dbl>          <dbl>  <dbl>      <dbl>     <dbl>    <dbl>
## 1     0.930        0.911  80.4      48.5 0.00000124     3
## # ... with 6 more variables: logLik <dbl>, AIC <dbl>,
## #   BIC <dbl>, deviance <dbl>, df.residual <int>,
## #   nobs <int>
```

```
broom::glance(lm_tm)
```

```
## # A tibble: 1 × 12
##   r.squared adj.r.squared sigma statistic   p.value     df
##       <dbl>          <dbl>  <dbl>      <dbl>     <dbl>    <dbl>
## 1     0.930        0.911  80.4      48.5 0.00000124     3
## # ... with 6 more variables: logLik <dbl>, AIC <dbl>,
## #   BIC <dbl>, deviance <dbl>, df.residual <int>,
## #   nobs <int>
```

Tidy model prediction

```
broom::augment(lm_tm, new_data = books)
```

```
## # A tibble: 15 × 5
##       volume weight cover     .pred   .resid
##       <dbl>    <dbl> <fct>    <dbl>    <dbl>
## 1      885     800 hardback  836.  -35.6
## 2     1016     950 hardback  935.  14.6
## 3     1125    1050 hardback 1018.  31.6
## 4      239     350 hardback  344.   6.39
## 5      701     750 hardback  695.  54.5
## 6      641     600 hardback  650.  -49.8
## 7     1228    1075 hardback 1097. -21.8
## 8      412     250 paperback 324.  -73.9
## 9      953     700 paperback 695.   5.00
## 10     929     650 paperback 679.  -28.5
## 11    1492     975 paperback 1065. -89.7
## 12     419     350 paperback 329.  21.3
## 13    1010     950 paperback 734.  216.
## 14     595     425 paperback 449.  -24.5
## 15    1034     725 paperback 751.  -25.6
```

Putting it together

```
lm_tm %>%
  augment(
    new_data = tidyrr::expand_grid(
      volume = seq(0, 1500, by=5),
      cover = c("hardback", "paperback") %>% as.factor()
    )
  ) %>%
  rename(weight = .pred) %>%
  ggplot(aes(x = volume, y = weight, color = cover, group = cover)) +
  geom_line() +
  geom_point(data = books)
```

Why do we care?



```
show_engines("linear_reg")  
  
## # A tibble: 5 × 2  
##   engine mode  
##   <chr>  <chr>  
## 1 lm     regression  
## 2 glmnet regression  
## 3 stan   regression  
## 4 spark   regression  
## 5 keras   regression  
  
(bayes_tm = linear_reg() %>%  
  set_engine(  
    "stan",  
    prior_intercept = rstanarm::student_t(df = 1),  
    prior = rstanarm::student_t(df = 1),  
    seed = 1234  
  )  
)  
  
## Linear Regression Model Specification (regression)  
##  
## Engine-Specific Arguments:  
##   prior_intercept = rstanarm::student_t(df = 1)  
##   prior = rstanarm::student_t(df = 1)  
##   seed = 1234  
##  
## Computational engine: stan
```

Fitting with rstanarm

```
(bayes_tm = bayes_tm %>%  
  fit(weight ~ volume * cover, data = books)  
)  
  
## Warning: There were 19 divergent transitions after warmup. See  
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup  
## to find out why this is a problem and how to eliminate them.  
  
## Warning: Examine the pairs() plot to diagnose sampling problems  
  
## parsnip model object  
##  
## Fit time: 911ms  
## stan_glm  
##   family:      gaussian [identity]  
##   formula:    weight ~ volume * cover  
##   observations: 15  
##   predictors:  4  
## -----  
##                   Median MAD_SD  
## (Intercept)      98.2   58.8  
## volume          0.8    0.1  
## coverpaperback -0.3   3.5  
## volume:coverpaperback -0.2   0.1  
##  
## Auxiliary parameter(s):  
##   Median MAD_SD  
## sigma 84.9   17.8  
##  
## See ?details_linear_reg_stan for details within parsnip  
## * For help interpreting the printed output see ?print.stanreg
```

What was actually run?

```
linear_reg() %>%  
  set_engine(  
    "stan",  
    prior_intercept = rstanarm::student_t(df = 1),  
    prior = rstanarm::student_t(df = 1),  
    seed = 1234  
) %>%  
  translate()  
  
## Linear Regression Model Specification (regression)  
##  
## Engine-Specific Arguments:  
##   prior_intercept = rstanarm::student_t(df = 1)  
##   prior = rstanarm::student_t(df = 1)  
##   seed = 1234  
##  
## Computational engine: stan  
##  
## Model fit template:  
## rstanarm::stan_glm(formula = missing_arg(), data = missing_arg(),  
##   weights = missing_arg(), prior_intercept = rstanarm::student_t(df = 1),  
##   prior = rstanarm::student_t(df = 1), seed = 1234, family = stats::gaussian,  
##   refresh = 0)
```

Back to broom

```
broom::tidy(bayes_tm)
```

```
## Error in warn_on_stanreg(x): The supplied model object seems to be outputted from the rstanarm package. Tidi
```

```
broom.mixed::tidy(bayes_tm)
```

```
## # A tibble: 4 × 3
##   term            estimate std.error
##   <chr>          <dbl>     <dbl>
## 1 (Intercept)    98.2      58.8
## 2 volume         0.829     0.0733
## 3 coverpaperback -0.285     3.53
## 4 volume:coverpaperback -0.197     0.0510
```

```
broom.mixed::glance(bayes_tm)
```

```
## # A tibble: 1 × 4
##   algorithm  pss  nobs sigma
##   <chr>     <dbl> <int> <dbl>
## 1 sampling    4000     15  84.9
```

Augment

```
augment(bayes_tm, new_data=books)
```

```
## # A tibble: 15 × 5
##   volume weight cover     .pred   .resid
##   <dbl>    <dbl> <fct>    <dbl>    <dbl>
## 1     885     800 hardback  829.   -29.2
## 2    1016     950 hardback  938.   12.3
## 3    1125    1050 hardback 1028.   22.1
## 4     239     350 hardback  294.   55.5
## 5     701     750 hardback  677.   73.1
## 6     641     600 hardback  627.  -27.3
## 7    1228    1075 hardback 1113.  -38.2
## 8     412     250 paperback 355.  -105.
## 9     953     700 paperback 696.    4.16
## 10    929     650 paperback 681.  -30.7
## 11   1492     975 paperback 1036.  -60.8
## 12    419     350 paperback  359.  -9.05
## 13   1010     950 paperback  732.   218.
## 14    595     425 paperback  470.  -45.1
## 15   1034     725 paperback  747.  -21.9
```

Predictions

```
bayes_tm %>%
  augment(
    new_data = tidyrr::expand_grid(
      volume = seq(0, 1500, by=5),
      cover = c("hardback", "paperback") %>% as.factor()
    )
  ) %>%
  rename(weight = .pred) %>%
  ggplot(aes(x = volume, y = weight, color = cover, group = cover)) +
  geom_line() +
  geom_point(data = books)
```

Cross validation and Feature engineering

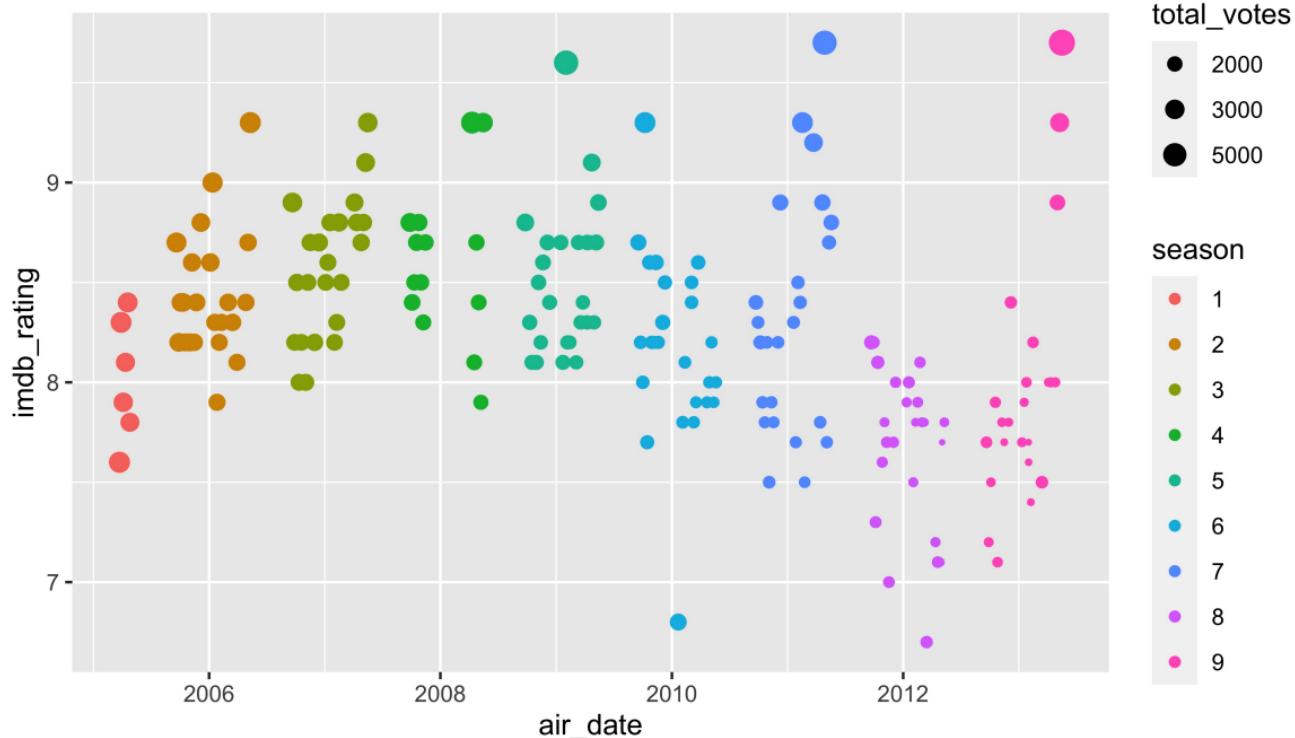
The Office & IMDB

```
(office_ratings = read_csv("data/office_ratings.csv"))
```

```
## # A tibble: 188 × 6
##   season episode title    imdb_rating total_votes air_date
##   <dbl>    <dbl> <chr>        <dbl>      <dbl> <date>
## 1     1       1 Pilot        7.6        3706 2005-03-24
## 2     1       2 Divers...     8.3        3566 2005-03-29
## 3     1       3 Health...     7.9        2983 2005-04-05
## 4     1       4 The Al...     8.1        2886 2005-04-12
## 5     1       5 Basket...     8.4        3179 2005-04-19
## 6     1       6 Hot Gi...     7.8        2852 2005-04-26
## 7     2       1 The Du...     8.7        3213 2005-09-20
## 8     2       2 Sexual...     8.2        2736 2005-09-27
## 9     2       3 Office...     8.4        2742 2005-10-04
## 10    2       4 The Fi...     8.4        2713 2005-10-11
## # ... with 178 more rows
```

These data are from [data.world](#), by way of [TidyTuesday](#).

Rating vs Air Date



Test-train split



```
set.seed(123)
(office_split = initial_split(office_ratings, pro
[| ]| )
## <Analysis/Assess/Total>
## <150/38/188>
```

```
(office_train = training(office_split))
```

```
## # A tibble: 150 x 6
##   season episode title    imdb_rating total_votes
##   <dbl>     <dbl> <chr>          <dbl>        <dbl>
## 1     8       18 Last D...      7.8         14
## 2     9       14 Vandal...      7.6         14
## 3     2       8 Perfor...      8.2         24
## 4     9       5 Here C...      7.1         15
## 5     3       22 Beach ...     9.1         27
## 6     7       1 Nepoti...      8.4         18
## 7     3       15 Phylli...      8.3         22
## 8     9       21 Livin'...      8.9         20
## 9     9       18 Promos        8           14
```

```
(office_test = testing(office_split))
```

```
## # A tibble: 38 × 6
##   season episode title    imdb_rating total_votes ai
##   <dbl>     <dbl> <chr>          <dbl>        <dbl> <d
## 1       1         1 Divers...      8.3        3566 20
## 2       2         2 The Fi...      8.4        2713 20
## 3       2         9 E-Mail...      8.4        2527 20
## 4       2        12 The In...      9          3282 20
## 5       2        22 Casino...     9.3        3644 20
## 6       3         5 Initia...     8.2        2254 20
## 7       3        16 Busine...     8.8        2622 20
## 8       3        17 Cockta...     8.5        2264 20
## 9       4         6 Branch...     8.5        2185 20
```

Feature engineering with dplyr

```
office_train %>%  
  mutate(  
    season = as_factor(season),  
    month = lubridate::month(air_date),  
    wday = lubridate::wday(air_date),  
    top10_votes = as.integer(total_votes > quantile(total_votes, 0.9))  
)
```

```
## # A tibble: 150 × 9  
##   season episode title           imdb_rating total_votes air_date   month   wday top10_votes  
##   <fct>     <dbl> <chr>            <dbl>        <dbl> <date>      <dbl> <dbl>       <int>  
## 1 8          18 Last Day in Florida     7.8        1429 2012-03-08     3     5         0  
## 2 9          14 Vandalism                 7.6        1402 2013-01-31     1     5         0  
## 3 2          8 Performance Review       8.2        2416 2005-11-15    11     3         0  
## 4 9          5 Here Comes Treble       7.1        1515 2012-10-25    10     5         0  
## 5 3          22 Beach Games                9.1        2783 2007-05-10     5     5         0  
## 6 7          1 Nepotism                  8.4        1897 2010-09-23    9     5         0  
## 7 3          15 Phyllis' Wedding        8.3        2283 2007-02-08     2     5         0  
## 8 9          21 Livin' the Dream        8.9        2041 2013-05-02     5     5         0  
## 9 9          18 Promos                   8          1445 2013-04-04     4     5         0  
## 10 8         12 Pool Party                8          1612 2012-01-19     1     5         0  
## # ... with 140 more rows
```

Better living through recipes



```
(r = recipe(imdb_rating ~ ., data = office_train)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##   outcome          1
## predictor         5
```

```
summary(r)
```

```
## # A tibble: 6 × 4
##   variable    type    role     source
##   <chr>      <chr>   <chr>    <chr>
## 1 season     numeric predictor original
## 2 episode    numeric predictor original
## 3 title      nominal predictor original
## 4 total_votes numeric predictor original
## 5 air_date    date    predictor original
## 6 imdb_rating numeric outcome  original
```

Recipe roles

```
(r = recipe(imdb_rating ~ ., data = office_train)
  update_role(title, new_role = "ID")
)

## Recipe
##
## Inputs:
##
##       role #variables
##       ID      1
##   outcome      1
## predictor      4
```

```
summary(r)
```

```
## # A tibble: 6 × 4
##   variable     type    role    source
##   <chr>        <chr>   <chr>   <chr>
## 1 season      numeric predictor original
## 2 episode     numeric predictor original
## 3 title       nominal  ID        original
## 4 total_votes numeric predictor original
## 5 air_date    date     predictor original
## 6 imdb_rating numeric outcome  original
```

Adding features (month & day of week)

```
(r = recipe(imdb_rating ~ ., data = office_train)
  update_role(title, new_role = "ID") %>%
  step_date(air_date, features = c("dow", "month")
)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##       ID      1
##   outcome      1
## predictor      4
##
## Operations:
##
## Date features from air_date
```

```
summary(r)
```

```
## # A tibble: 6 × 4
##   variable     type    role    source
##   <chr>       <chr>   <chr>   <chr>
## 1 season      numeric predictor original
## 2 episode     numeric predictor original
## 3 title       nominal  ID        original
## 4 total_votes numeric predictor original
## 5 air_date    date     predictor original
## 6 imdb_rating numeric outcome  original
```

Adding Holidays

```
(r = recipe(imdb_rating ~ ., data = office_train) %>%
  update_role(title, new_role = "ID") %>%
  step_date(air_date, features = c("dow", "month")) %>%
  step_holiday(
    air_date,
    holidays = c("USThanksgivingDay", "USChristmasDay", "USNewYearsDay", "USIndependenceDay"),
    keep_original_cols = FALSE
  )
)
```

```
## Recipe
##
## Inputs:
##
##     role #variables
##         ID          1
##     outcome          1
## predictor          4
##
## Operations:
##
## Date features from air_date
## Holiday features from air_date
```

Seasons as factors

```
(r = recipe(imdb_rating ~ ., data = office_train) %>%
  update_role(title, new_role = "ID") %>%
  step_date(air_date, features = c("dow", "month")) %>%
  step_holiday(
    air_date,
    holidays = c("USThanksgivingDay", "USChristmasDay", "USNewYearsDay", "USIndependenceDay"),
    keep_original_cols = FALSE
  ) %>%
  step_num2factor(season, levels = as.character(1:9))
)
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##       ID          1
##   outcome          1
## predictor         4
##
## Operations:
##
## Date features from air_date
## Holiday features from air_date
```

Dummy coding

```
(r = recipe(imdb_rating ~ ., data = office_train) %>%
  update_role(title, new_role = "ID") %>%
  step_date(air_date, features = c("dow", "month")) %>%
  step_holiday(
    air_date,
    holidays = c("USThanksgivingDay", "USChristmasDay", "USNewYearsDay", "USIndependenceDay"),
    keep_original_cols = FALSE
  ) %>%
  step_num2factor(season, levels = as.character(1:9)) %>%
  step_dummy(all_nominal_predictors())
)
```

```
## Recipe
##
## Inputs:
##
##      role #variables
##      ID          1
##      outcome       1
##      predictor     4
##
## Operations:
##
## Date features from air_date
```

Why no `top10_votes`?

There does not appear to be a `step_*`() function that implements the quantile calculation necessary for the `top10_votes` feature above, custom recipe steps can be created see [this](#) for an relevant example.

Preparing a recipe

```
prep(r)
```

```
## Recipe
##
## Inputs:
##
##      role #variables
##      ID          1
##      outcome       1
##      predictor     4
##
## Training data contained 150 data points and no missing data.
##
## Operations:
##
## Date features from air_date [trained]
## Holiday features from air_date [trained]
## Factor variables from season [trained]
## Dummy variables from season, air_date_dow, air_date_month [trained]
```

Baking a recipe

```
prep(r) %>%  
  bake(new_data = office_train)  
  
## # A tibble: 150 × 33  
##   episode title    total_votes  imdb_rating air_date_USThanksg... air_date_USChris... air_date_USNewYe...  
##   <dbl> <fct>        <dbl>        <dbl>            <dbl>            <dbl>            <dbl>  
## 1     18 Last Day...     1429       7.8             0             0             0  
## 2     14 Vandalism      1402       7.6             0             0             0  
## 3      8 Performa...     2416       8.2             0             0             0  
## 4      5 Here Com...     1515       7.1             0             0             0  
## 5     22 Beach Ga...     2783       9.1             0             0             0  
## 6      1 Nepotism       1897       8.4             0             0             0  
## 7     15 Phyllis'...     2283       8.3             0             0             0  
## 8     21 Livin' t...     2041       8.9             0             0             0  
## 9     18 Promos         1445        8             0             0             0  
## 10    12 Pool Par...     1612        8             0             0             0  
## # ... with 140 more rows, and 26 more variables: air_date_USIndependenceDay <dbl>, season_X2 <dbl>,  
## #   season_X3 <dbl>, season_X4 <dbl>, season_X5 <dbl>, season_X6 <dbl>, season_X7 <dbl>,  
## #   season_X8 <dbl>, season_X9 <dbl>, air_date_dow_Mon <dbl>, air_date_dow_Tue <dbl>,  
## #   air_date_dow_Wed <dbl>, air_date_dow_Thu <dbl>, air_date_dow_Fri <dbl>, air_date_dow_Sat <dbl>,  
## #   air_date_month_Feb <dbl>, air_date_month_Mar <dbl>, air_date_month_Apr <dbl>,  
## #   air_date_month_May <dbl>, air_date_month_Jun <dbl>, air_date_month_Jul <dbl>,  
## #   air_date_month_Aug <dbl>, air_date_month_Sep <dbl>, air_date_month_Oct <dbl>, ...
```

Informative features?

```
prep(r) %>%  
  bake(new_data = office_train) %>%  
  map_int(~ length(unique(.x)))
```

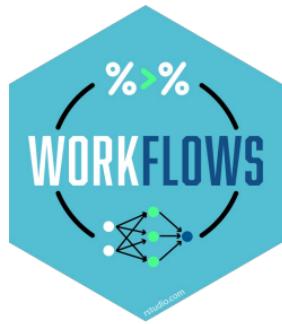
```
##          episode           title      total_votes  
##          26             150                 142  
##  imdb_rating air_date_USThanksgivingDay air_date_USChristmasDay  
##          26                         1                   1  
## air_date_USNewYearsDay air_date_USIndependenceDay season_X2  
##          1                         1                   2  
##          season_X3           season_X4      season_X5  
##          2                         2                   2  
##          season_X6           season_X7      season_X8  
##          2                         2                   2  
##          season_X9           air_date_dow_Mon air_date_dow_Tue  
##          2                         1                   2  
##          air_date_dow_Wed       air_date_dow_Thu air_date_dow_Fri  
##          1                         2                   1  
##          air_date_dow_Sat       air_date_month_Feb air_date_month_Mar  
##          1                         2                   2  
##          air_date_month_Apr      air_date_month_May air_date_month_Jun  
##          2                         2                   1  
##          air_date_month_Jul      air_date_month_Aug air_date_month_Sep  
##          1                         1                   2
```

Removing zero variance predictors

```
r = recipe(imdb_rating ~ ., data = office_train) %>%  
  update_role(title, new_role = "ID") %>%  
  step_date(air_date, features = c("dow", "month")) %>%  
  step_holiday(  
    air_date,  
    holidays = c("USThanksgivingDay", "USChristmasDay", "USNewYearsDay", "USIndependenceDay"),  
    keep_original_cols = FALSE  
) %>%  
  step_num2factor(season, levels = as.character(1:9)) %>%  
  step_dummy(all_nominal_predictors()) %>%  
  step_zv(all_predictors())  
  
prep(r) %>%  
  bake(new_data = office_train)
```

```
## # A tibble: 150 × 22  
##   episode title total_votes imdb_rating season_X2 season_X3 season_X4 season_X5 season_X6 season_X7  
##   <dbl> <fct>     <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>  
## 1     18 Last...     1429       7.8        0         0         0         0         0         0  
## 2     14 Vand...     1402       7.6        0         0         0         0         0         0  
## 3      8 Perf...     2416       8.2        1         0         0         0         0         0  
## 4      5 Here...     1515       7.1        0         0         0         0         0         0  
## 5     22 Beac...     2783       9.1        0         1         0         0         0         0  
## 6      1 Nepo...     1897       8.4        0         0         0         0         0         1  
## 7     15 Phyl...     2283       8.3        0         1         0         0         0         0  
## 8     21 Livi...     2041       8.9        0         0         0         0         0         0  
## 9     18 Prom...     1445        8        0         0         0         0         0         0  
## 10    12 Pool...     1612        8        0         0         0         0         0         0  
## # ... with 140 more rows, and 12 more variables: season_X8 <dbl>, season_X9 <dbl>,  
## #   air_date_dow_Tue <dbl>, air_date_dow_Thu <dbl>, air_date_month_Feb <dbl>,
```

Really putting it all together



```
(office_work = workflow() %>%  
  add_recipe(r) %>%  
  add_model(  
    linear_reg() %>%  
    set_engine("lm")  
  )  
)  
  
## == Workflow =====  
## Preprocessor: Recipe  
## Model: linear_reg()  
##  
## — Preprocessor -----  
## 5 Recipe Steps  
##  
## • step_date()  
## • step_holiday()  
## • step_num2factor()  
## • step_dummy()  
## • step_zv()  
##  
## — Model -----  
## Linear Regression Model Specification (regressi
```

Workflow fit

```
(office_fit = office_work %>%
  fit(data = office_train))

## == Workflow [trained] -----
## Preprocessor: Recipe
## Model: linear_reg()
##
## — Preprocessor -----
## 5 Recipe Steps
##
## • step_date()
## • step_holiday()
## • step_num2factor()
## • step_dummy()
## • step_zv()
##
## — Model -----
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
## (Intercept)      episode      total_votes    season_X2    season_X3
##       6.2557856   -0.0081872    0.0003838    0.8389817   1.1277917
##   season_X4      season_X5      season_X6    season_X7    season_X8
##       1.1519446    1.1767757    1.0899544    1.0671645   0.5024513
##   season_X9 air_date_dow_Tue air_date_dow_Thu air_date_month_Feb air_date_month_Mar
##       0.6981099     0.4945405    0.3699592    -0.0089843   -0.0194673
## air_date_month_Apr air_date_month_May air_date_month_Sep air_date_month_Oct air_date_month_Nov
##       0.0895888     0.2105803    -0.0776988    -0.2059183   -0.1895729
```

Performance

```
office_fit %>%
  augment(office_train) %>%
  rmse(imdb_rating, .pred)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>       <dbl>
## 1 rmse    standard     0.305
```

```
office_fit %>%
  augment(office_test) %>%
  rmse(imdb_rating, .pred)
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>       <dbl>
## 1 rmse    standard     0.423
```

k-fold cross validation

	training					testing
	validate	train	train	train	train	
fold 1	validate	train	train	train	train	
fold 2	train	validate	train	train	train	
fold 3	train	train	validate	train	train	
fold 4	train	train	train	validate	train	
fold 5	train	train	train	train	validate	

Creating folds

```
set.seed(123)
(folds = vfold_cv(office_train, v=5))

## # 5-fold cross-validation
## # A tibble: 5 × 2
##   splits          id
##   <list>        <chr>
## 1 <split [120/30]> Fold1
## 2 <split [120/30]> Fold2
## 3 <split [120/30]> Fold3
## 4 <split [120/30]> Fold4
## 5 <split [120/30]> Fold5

(office_fit_folds = office_work %>%
  fit_resamples(folds)
)

## ! Fold2: preprocessor 1/1, model 1/1 (predictions): prediction from a rank-deficient fit may be misle...
## Warning: This tuning result has notes. Example notes on model fitting include:
## preprocessor 1/1, model 1/1 (predictions): prediction from a rank-deficient fit may be misleading

## # Resampling results
## # 5-fold cross-validation
```

Fold performance

```
tune::collect_metrics(office_fit_folds)

## # A tibble: 2 × 6
##   .metric .estimator  mean    n std_err .config
##   <chr>   <chr>     <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.421     5  0.0511 Preprocessor1_Model1
## 2 rsq     standard   0.550     5  0.0663 Preprocessor1_Model1

tune::collect_metrics(office_fit_folds, summarize = FALSE) %>%
  filter(.metric == "rmse")

## # A tibble: 5 × 5
##   id    .metric .estimator .estimate .config
##   <chr> <chr>   <chr>      <dbl> <chr>
## 1 Fold1 rmse    standard     0.618 Preprocessor1_Model1
## 2 Fold2 rmse    standard     0.382 Preprocessor1_Model1
## 3 Fold3 rmse    standard     0.321 Preprocessor1_Model1
## 4 Fold4 rmse    standard     0.410 Preprocessor1_Model1
## 5 Fold5 rmse    standard     0.377 Preprocessor1_Model1
```

More on the `tune` package next time