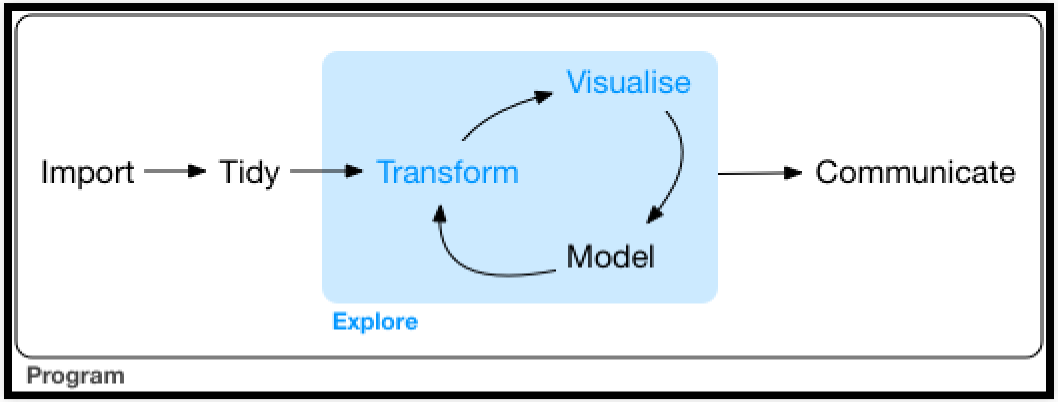
Introduction to the tidyverse: Data analysis with `infer` and `broom`

Christopher Skovron
Northwestern University

July 18, 2018

The workflow: modeling



How do I do tidy data analysis?

Everything we've seen so far has been relatively complete. The final part of the tidyverse that's under the most development is modeling.

Two packages that help so far

- infer, for tidy hypothesis tests
- broom, to tidy regression output

Data preparation

```r
library(nycflights13)
library(tidyverse)
library(infer)
set.seed(2017)
fli_small <- flights %>%
  sample_n(size = 500) %>%
  mutate(half_year = case_when(
    between(month, 1, 6) ~ "h1",
    between(month, 7, 12) ~ "h2"
  )) %>%
  mutate(day_hour = case_when(
    between(hour, 1, 12) ~ "morning",
    between(hour, 13, 24) ~ "not morning"
  )) %>%
  select(arr_delay, dep_delay, half_year,
         day_hour, origin, carrier)
```

Data preparation

- Two numeric - `arr_delay`, `dep_delay`
- Two categories
  - `half_year` (`"h1"`, `"h2"`),
  - `day_hour` (`"morning"`, `"not morning"`)
- Three categories - `origin` (`"EWR"`, `"JFK"`, `"LGA"`)
- Sixteen categories - `carrier`

Calculate observed statistic

The recommended approach is to use `specify() %>% calculate()`:

```
obs_t <- fli_small %>%
  specify(arr_delay ~ half_year) %>%
  calculate(stat = "t", order = c("h1", "h2"))
```

```
## Warning: Removed 15 rows containing missing values.
```

The observed t statistic is 0.8685463.

Calculate observed statistic

Or using `t_test` in `infer`

```
obs_t <- fli_small %>%
  t_test(formula = arr_delay ~ half_year, alternative = "two_sided",
         order = c("h1", "h2")) %>%
  dplyr::select(statistic) %>%
  dplyr::pull()
```

The observed t statistic is 0.8685463.

Calculate observed statistic

Or using another shortcut function in `infer`:

```
obs_t <- fli_small %>%
  t_stat(formula = arr_delay ~ half_year, order = c("h1", "h2"))
```

```
## Warning: Removed 15 rows containing missing values.
```

The observed t statistic is 0.8685463.

Data preparation - chi-square

```
library(nycflights13)
library(tidyverse)
library(infer)
set.seed(2017)
fli_small <- flights %>%
  na.omit() %>%
  sample_n(size = 500) %>%
  mutate(season = case_when(
    month %in% c(10:12, 1:3) ~ "winter",
    month %in% c(4:9) ~ "summer"
  )) %>%
  mutate(day_hour = case_when(
    between(hour, 1, 12) ~ "morning",
    between(hour, 13, 24) ~ "not morning"
  )) %>%
  select(arr_delay, dep_delay, season,
         day_hour, origin, carrier)
```

Calculate observed statistic

The recommended approach is to use `specify() %>% calculate()`:

```
obs_chisq <- fli_small %>%
  specify(origin ~ season) %>% # alt: response = origin, explanatory = se
  calculate(stat = "Chisq")
```

The observed χ^2 statistic is 0.571898.

Calculate observed statistic

Or using `chisq_test` in `infer`

```
obs_chisq <- fli_small %>%
  chisq_test(formula = origin ~ season) %>%
  dplyr::select(statistic)
```

Again, the observed χ^2 statistic is 0.571898.

Calculate observed statistic

Or using another shortcut function in `infer`:

```
obs_chisq <- fli_small %>%
  chisq_stat(formula = origin ~ season)
```

Lastly, the observed χ^2 statistic is 0.571898.

broom: let's tidy up a bit

The broom package takes the messy output of built-in functions in R, such as `lm`, `nls`, or `t.test`, and turns them into tidy data frames.

Tidying functions

- `tidy`: constructs a data frame that summarizes the model's statistical findings. This includes coefficients and p-values for each term in a regression, per-cluster information in clustering applications, or per-test information for `multtest` functions.
- `augment`: add columns to the original data that was modeled. This includes predictions, residuals, and cluster assignments.
- `glance`: construct a concise *one-row* summary of the model. This typically contains values such as R^2, adjusted R^2, and residual standard error that are computed once for the entire model.

Example

```r
lmfit <- lm(mpg ~ wt, mtcars)
lmfit
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)           wt
##      37.285       -5.344
```

```r
summary(lmfit)
```

```
##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
```

```
## (Intercept)  37.2851     1.8776  19.858  < 2e-16 ***
## wt           -5.3445     0.5591  -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

Tidy the output

Instead, you can use the `tidy` function, from the broom package, on the fit:

```
library(broom)
tidy(lmfit)
```

```
##          term  estimate std.error statistic      p.value
## 1 (Intercept) 37.285126  1.877627 19.857575 8.241799e-19
## 2          wt -5.344472  0.559101 -9.559044 1.293959e-10
```

Tidy the output

This gives you a data.frame representation. Note that the row names have been moved into a column called `term`, and the column names are simple and consistent (and can be accessed using `$`).

Augment the output

Instead of viewing the coefficients, you might be interested in the fitted values and residuals for each of the original points in the regression. For this, use `augment`, which augments the original data with information from the model:

```
augment(lmfit)
```

```
##              .rownames  mpg    wt  .fitted   .se.fit     .resid
## 1            Mazda RX4 21.0 2.620 23.282611 0.6335798 -2.2826106
## 2        Mazda RX4 Wag 21.0 2.875 21.919770 0.5714319 -0.9197704
## 3           Datsun 710 22.8 2.320 24.885952 0.7359177 -2.0859521
## 4       Hornet 4 Drive 21.4 3.215 20.102650 0.5384424  1.2973499
## 5    Hornet Sportabout 18.7 3.440 18.900144 0.5526562 -0.2001440
## 6              Valiant 18.1 3.460 18.793255 0.5552829 -0.6932545
## 7           Duster 360 14.3 3.570 18.205363 0.5734244 -3.9053627
## 8            Merc 240D 24.4 3.190 20.236262 0.5386565  4.1637381
## 9             Merc 230 22.8 3.150 20.450041 0.5397522  2.3499593
## 10            Merc 280 19.2 3.440 18.900144 0.5526562  0.2998560
## 11           Merc 280C 17.8 3.440 18.900144 0.5526562 -1.1001440
## 12          Merc 450SE 16.4 4.070 15.533127 0.7191881  0.8668731
## 13          Merc 450SL 17.3 3.730 17.350247 0.6100029 -0.0502472
## 14         Merc 450SLC 15.2 3.780 17.083024 0.6236291 -1.8830236
```

```
## 15  Cadillac Fleetwood 10.4 5.250  9.226650 1.2576087  1.1733496
## 16 Lincoln Continental 10.4 5.424  8.296712 1.3461693  2.1032876
```

Note that each of the new columns begins with a `.` (to avoid overwriting any of the original columns).

Glance at the output

Finally, several summary statistics are computed for the entire regression, such as R^2 and the F-statistic. These can be accessed with the `glance` function:

```
glance(lmfit)
```

```
##   r.squared adj.r.squared    sigma statistic      p.value df    logLil
## 1 0.7528328     0.7445939 3.045882  91.37533 1.293959e-10  2 -80.0147
##        AIC      BIC deviance df.residual
## 1 166.0294 170.4266 278.3219          30
```

Generalized linear and non-linear models

These functions apply equally well to the output from `glm`:

```
glmfit <- glm(am ~ wt, mtcars, family="binomial")
tidy(glmfit)
```

```
##          term estimate std.error statistic     p.value
## 1 (Intercept) 12.04037  4.509706  2.669879 0.007587858
## 2          wt -4.02397  1.436416 -2.801396 0.005088198
```

```
augment(glmfit)
```

```
##               .rownames am    wt    .fitted   .se.fit      .resid
## 1             Mazda RX4  1 2.620  1.4975684 0.9175750  0.63538540
## 2         Mazda RX4 Wag  1 2.875  0.4714561 0.6761141  0.98483443
## 3            Datsun 710  1 2.320  2.7047594 1.2799233  0.35984584
## 4        Hornet 4 Drive  0 3.215 -0.8966937 0.6012064 -0.82717675
## 5     Hornet Sportabout  0 3.440 -1.8020869 0.7486164 -0.55259722
## 6               Valiant  0 3.460 -1.8825663 0.7669573 -0.53230123
## 7            Duster 360  0 3.570 -2.3252030 0.8778451 -0.43191437
## 8             Merc 240D  0 3.190 -0.7960945 0.5934948 -0.86291728
## 9              Merc 230  0 3.150 -0.6351357 0.5855423 -0.92214813
## 10             Merc 280  0 3.440 -1.8020869 0.7486164 -0.55259722
## 11            Merc 280C  0 3.440 -1.8020869 0.7486164 -0.55259722
```

```
## 12          Merc 450SE  0 4.070 -4.3371880 1.4929683 -0.16117395
## 13          Merc 450SL  0 3.730 -2.9690382 1.0606783 -0.31647306
## 14         Merc 450SLC  0 3.780 -3.1702367 1.1213659 -0.28683046
## 15  Cadillac Fleetwood  0 5.250 -9.0854725 3.1242454 -0.01505280
## 16 Lincoln Continental  0 5.424 -9.7856433 3.3701122 -0.01060678
```

```r
glance(glmfit)
```

```
##   null.deviance df.null    logLik      AIC      BIC deviance df.resid
## 1      43.22973      31 -9.588042 23.17608 26.10756 19.17608
```

Generalized linear and non-linear models

These functions also work on other fits, such as nonlinear models (`nls`):

```
nlsfit <- nls(mpg ~ k / wt + b, mtcars, start=list(k=1, b=0))
tidy(nlsfit)
```

```
##   term  estimate std.error statistic      p.value
## 1    k 45.829488  4.249155 10.785554 7.639162e-12
## 2    b  4.386254  1.536418  2.854858 7.737378e-03
```

```
augment(nlsfit, mtcars)
```

```
##                .rownames  mpg cyl  disp  hp drat    wt  qsec vs am gear
## 1              Mazda RX4 21.0   6 160.0 110 3.90 2.620 16.46  0  1    4
## 2          Mazda RX4 Wag 21.0   6 160.0 110 3.90 2.875 17.02  0  1    4
## 3             Datsun 710 22.8   4 108.0  93 3.85 2.320 18.61  1  1    4
## 4         Hornet 4 Drive 21.4   6 258.0 110 3.08 3.215 19.44  1  0    3
## 5      Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3
## 6                Valiant 18.1   6 225.0 105 2.76 3.460 20.22  1  0    3
## 7             Duster 360 14.3   8 360.0 245 3.21 3.570 15.84  0  0    3
## 8              Merc 240D 24.4   4 146.7  62 3.69 3.190 20.00  1  0    4
## 9               Merc 230 22.8   4 140.8  95 3.92 3.150 22.90  1  0    4
```

```
## 10             Merc 280 19.2   6 167.6 123 3.92 3.440 18.30  1  0    4
## 11            Merc 280C 17.8   6 167.6 123 3.92 3.440 18.90  1  0    4
## 12           Merc 450SE 16.4   8 275.8 180 3.07 4.070 17.40  0  0    3
## 13           Merc 450SL 17.3   8 275.8 180 3.07 3.730 17.60  0  0    3
## 14          Merc 450SLC 15.2   8 275.8 180 3.07 3.780 18.00  0  0    3
## 15   Cadillac Fleetwood 10.4   8 472.0 205 2.93 5.250 17.98  0  0    3
## 16  Lincoln Continental 10.4   8 460.0 215 3.00 5.424 17.82  0  0    3
```

```
glance(nlsfit)
```

```
##     sigma isConv      finTol    logLik      AIC      BIC deviance
## 1 2.77405   TRUE 2.87694e-08 -77.02329 160.0466 164.4438 230.8606
##   df.residual
## 1          30
```

Hypothesis testing

The `tidy` function can also be applied to `htest` objects, such as those output by popular built-in functions like `t.test`, `cor.test`, and `wilcox.test`.

```
tt <- t.test(wt ~ am, mtcars)
tidy(tt)
```

```
##   estimate estimate1 estimate2 statistic     p.value parameter  conf.l
## 1 1.357895  3.768895     2.411  5.493905 6.27202e-06  29.23352 0.85256
##   conf.high                  method alternative
## 1  1.863226 Welch Two Sample t-test   two.sided
```

Hypothesis testing

Some cases might have fewer columns (for example, no confidence interval):

```
wt <- wilcox.test(wt ~ am, mtcars)
tidy(wt)
```

```
##   statistic      p.value                                              me
## 1     230.5 4.347026e-05 Wilcoxon rank sum test with continuity correc
##   alternative
## 1   two.sided
```

Hypothesis testing

Since the `tidy` output is already only one row, `glance` returns the same output:

```
glance(tt)
```

```
##   estimate estimate1 estimate2 statistic      p.value parameter  conf.l
## 1 1.357895  3.768895     2.411  5.493905 6.27202e-06  29.23352 0.8525
##   conf.high                  method alternative
## 1  1.863226 Welch Two Sample t-test   two.sided
```

```
glance(wt)
```

```
##   statistic      p.value                                              me
## 1     230.5 4.347026e-05 Wilcoxon rank sum test with continuity correc
##   alternative
## 1   two.sided
```