TextLink 2017 - explor_pdt30

Silvie Cinkova

27 January 2017

```
library(dplyr)
library(tidyr)
library(stringr)
library(ggplot2)
library(ggthemes)
library(plotluck)
library(scales)
library(formatR)
```

Data Structure

```
pdt30 <- readRDS("src_data/pdt_30.RDS")
```

Data Set at First Sight

```
dplyr::glimpse(pdt30)
```

```
## Observations: 20,556
## Variables: 6
## $ document_id         <fctr> cmpr9410_001, cmpr9410_001, cmpr9410_001,...
## $ genre               <fctr> comment, comment, comment, comment, comme...
## $ number_of_sentences <int> 33, 33, 33, 33, 33, 33, 33, 33, 33, 33, 33...
## $ sentence_id         <fctr> t-cmpr9410-001-p10s2, t-cmpr9410-001-p11s...
## $ discourse_type      <fctr> conc, opp, spec, restr, conj, conc, restr...
## $ discourse_class     <fctr> CONTRAST, CONTRAST, EXPANSION, CONTRAST, ...
```

Interpretation: 5 categorical variables, 1 numerical variable

```
summary(pdt30)
```

```
##       document_id             genre      number_of_sentences
##  ln94207_73 :  123   news        :4510   Min.   :  1.0
##  ln94207_76 :  118   essay       :3757   1st Qu.: 16.0
##  ln94210_95 :  102   sport       :2525   Median : 26.0
##  ln94211_92 :   87   description :2305   Mean   : 37.3
##  ln94200_127:   80   comment     :1731   3rd Qu.: 42.0
##  ln94207_79 :   77   topic_interv:1247   Max.   :231.0
##  (Other)    :19969   (Other)     :4481
##                sentence_id    discourse_type    discourse_class
##  t-ln94203-125-p8s11 :    6   conj   :7498   CONTINGENCY:4701
##  t-ln94207-73-p13s7  :    6   opp    :3196   CONTRAST   :5938
##  t-ln94207-90-p4s9   :    6   reason :2632   EXPANSION  :8851
##  t-ln95048-061-p7s3  :    6   cond   :1369   TEMPORAL   :1066
```

```
##  t-cmpr9415-033-p11s4:    5   conc    : 880
##  t-ln94203-99-p8s1   :    5   preced : 840
##  (Other)             :20522   (Other):4141
```

Interpretation: Some levels of categorical variables and their frequency, summary of numerical variable

```
str(pdt30)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    20556 obs. of  6 variables:
##  $ document_id        : Factor w/ 2580 levels "cmpr9410_001",..: 1 1 1 1 1
1 1 1 1 1 ...
##  $ genre              : Factor w/ 19 levels "advice","caption",..: 4 4 4 4
4 4 4 4 4 4 ...
##  $ number_of_sentences: int  33 33 33 33 33 33 33 33 33 33 ...
##  $ sentence_id        : Factor w/ 16112 levels "t-cmpr9410-001-p10s2",..:
1 2 3 4 4 5 6 6 7 8 ...
##  $ discourse_type     : Factor w/ 23 levels "conc","cond",..: 1 16 22 21 4
1 21 4 2 16 ...
##  $ discourse_class    : Factor w/ 4 levels "CONTINGENCY",..: 2 2 3 2 3 2 2
3 1 2 ...
```

Interpretation: all levels of categorical variables

Visualization

Our PDT 3.0 Subcorpus

Why "subcorpus"? Our table records observations of connectives. Therefore we do not consider PDT30 documents where no connectives have occurred. (Maybe there are no such texts, but for the sake of precision!) Anyway, henceforth it will be called corpus.

How Many How Long Texts Are There in the Corpus?

```
doclen_set <- pdt30 %>% dplyr::distinct(document_id, .keep_all = TRUE) %>%
    dplyr::select(-c(starts_with("discourse"), starts_with("sentence")))
set.seed(122)
dplyr::sample_n(doclen_set, 10)
```

```
## # A tibble: 10 × 3
##     document_id         genre number_of_sentences
##           <fctr>        <fctr>               <int>
## 1  mf920925_116          news                   7
## 2  mf920925_120 person_interv                  52
## 3    ln94203_75   description                  25
## 4  cmpr9415_018       comment                  26
## 5   ln95045_059          news                   6
## 6   ln95046_078          news                   5
## 7   ln95047_120          news                  13
## 8  cmpr9410_008        advice                  64
```

```
## 9  cmpr9413_052          essay                    58
## 10  ln95045_110           news                    11
```

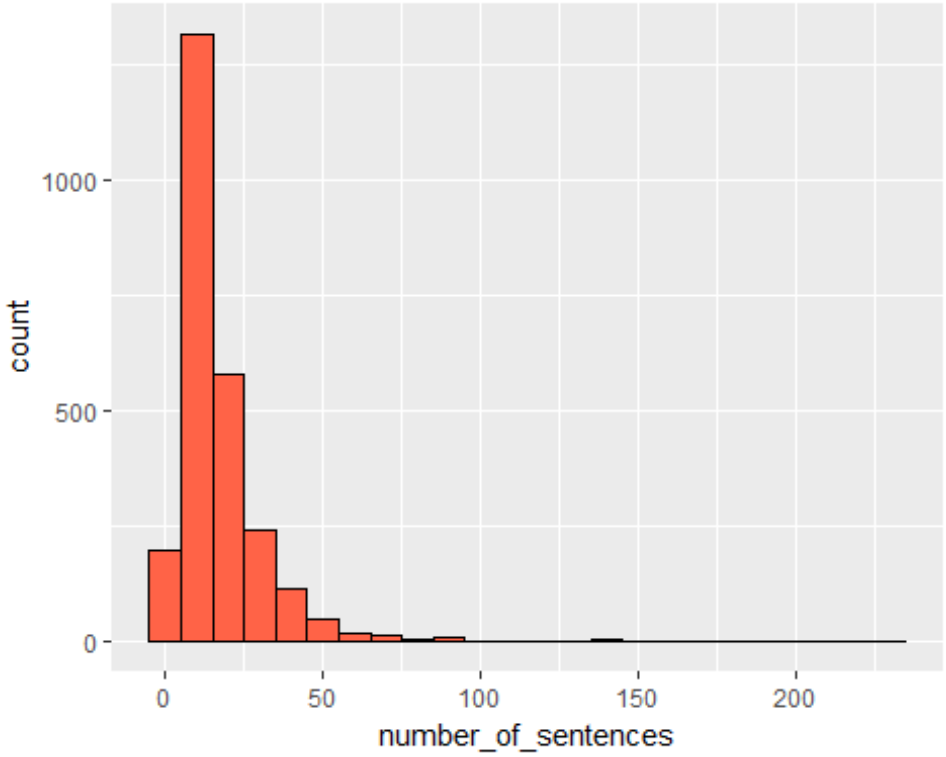
How many texts are there in the corpus?

```
nrow(doclen_set)

## [1] 2580
```

We are exploring a single variable: text length. Each text is one observation of this quantitative variable. A common diagram: *histogram*. X-axis: variable values, Y-axis: frequency of such. We see the *distribution of text lengths in the subcorpus*.

```
ggplot(doclen_set, aes(x = number_of_sentences)) + geom_histogram(binwidth =
10,
    col = "black", fill = "tomato")
```



The width of *bins* is adjustable. "Binwidth 10" means: the first bin contains texts of length 0-9 sentences, the second of 10-19 sentences, etc. There are somewhat less than 250 texts between 0 and 10 sentences long, 1300 texts between 10 and 20 sentences long, etc.There is a tiny number of texts (one?) over 150 sentences long.

How many texts do we actually have? As many as there are levels of the `document_id` factor in the `pdt30` data frame, see `str(pdt30)` above. But how many sentences are they long?

```
sentsums <- dplyr::summarise(group_by(doclen_set, genre), sum(number_of_sentences))
colnames(sentsums)[2] <- "sumsentnumbers"
sentsums
```

```
## # A tibble: 19 × 2
##            genre sumsentnumbers
##           <fctr>          <int>
## 1         advice           1501
## 2        caption             90
## 3     collection           1833
## 4        comment           3203
## 5    description           5850
## 6          essay           6793
## 7     invitation            693
## 8         letter            434
## 9           news          12537
## 10         other            974
## 11      overview            511
## 12 person_interv           1471
## 13          plot             73
## 14       program            146
## 15        review           2314
## 16         sport           4817
## 17        survey            355
## 18  topic_interv           2602
## 19       weather            105
```

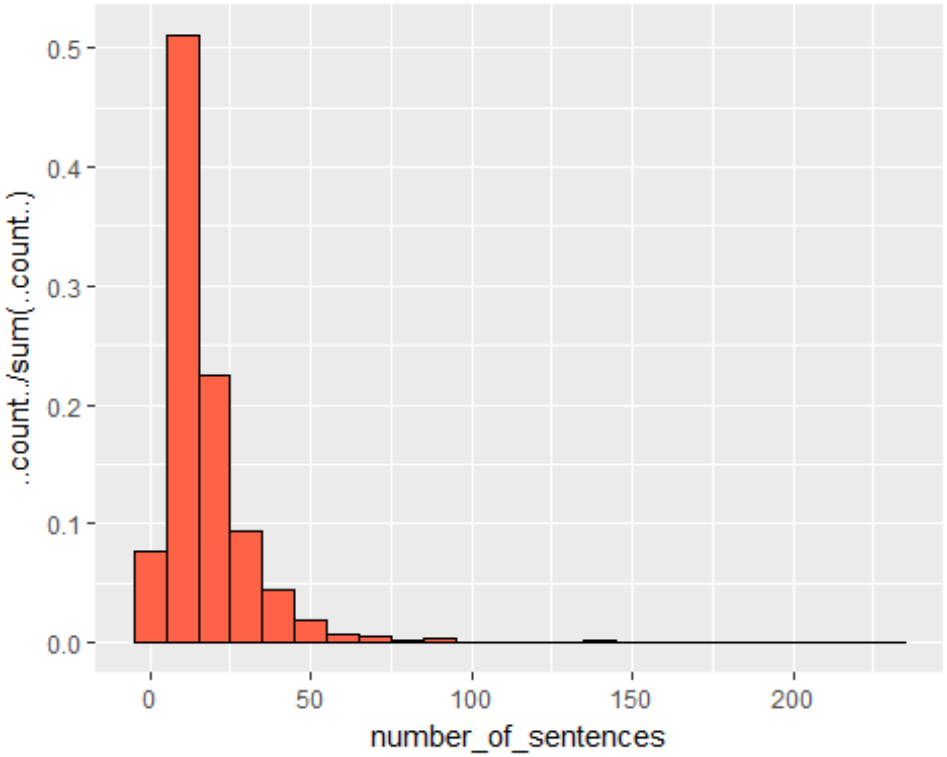
And how many sentences long is the entire corpus?

```
dplyr::summarise(doclen_set, sum(number_of_sentences))
```

```
## # A tibble: 1 × 1
##   `sum(number_of_sentences)`
##                        <int>
## 1                      46302
```

We may anyway also want to visualize the relative frequencies of different text lengths:

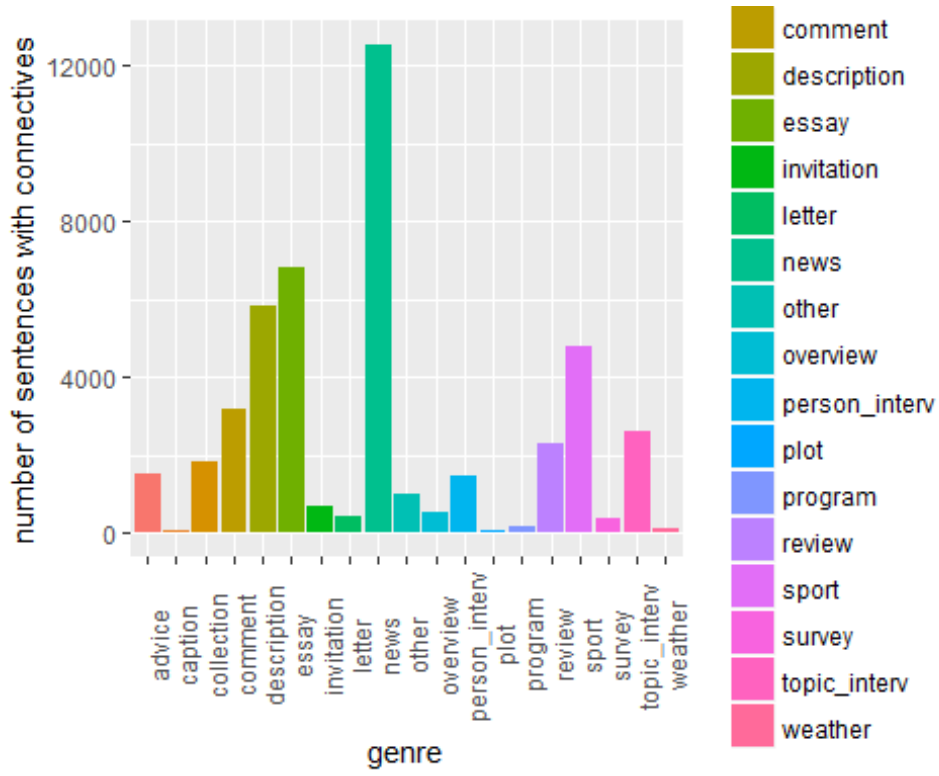
```
ggplot(doclen_set, aes(x = number_of_sentences)) + geom_histogram(aes(x = num
ber_of_sentences,
    y = ..count../sum(..count..)), col = "black", fill = "tomato",
    binwidth = 10)
```



How Much Text Is There in the Corpus for Each Genre?

Text is calculated in length, i.e. number of sentences. This time we focus on the text bulk in each genre, not distinguishing individual documents. We add color distinction to genres for easier comparison with the following plots, although the colors add no information to the barplot.

```
ggplot(sentsums, aes(y = sumsentnumbers, x = genre)) + geom_bar(stat = "ident
ity",
    aes(fill = genre)) + theme(axis.text.x = element_text(angle = 90)) +
    ylab("number of sentences with connectives")
```

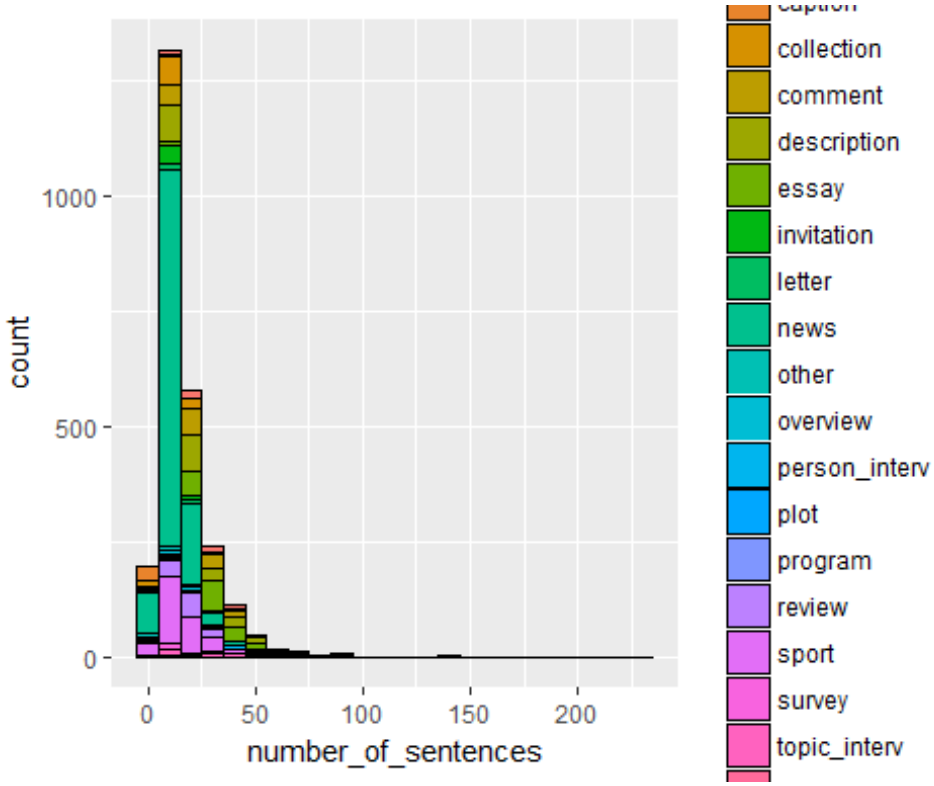


BTW - What is the difference between a histogram and a barplot?

How Many How Long Texts Are There in the Corpus for Each Genre? (I)

Normally, the distribution of sentence_lengths of documents would be all a histogram would be able to tell, since the y-axis always shows frequency or probability density (we ignore this one) and there would be only one axis left to project a variable onto. Thanks to ggplot2, however, we can add *genre* as a *second variable*, deploying *color fill*, which we can regard as a "third axis". We add lines for better color distinction.

```
# doclen_set <- pdt30 %>% distinct(document_id, .keep_all =
# TRUE) %>% select(-c(starts_with('discourse'),
# starts_with('sentence')))
ggplot(doclen_set, aes(x = number_of_sentences, fill = genre)) +
    geom_histogram(binwidth = 10, col = "black")
```

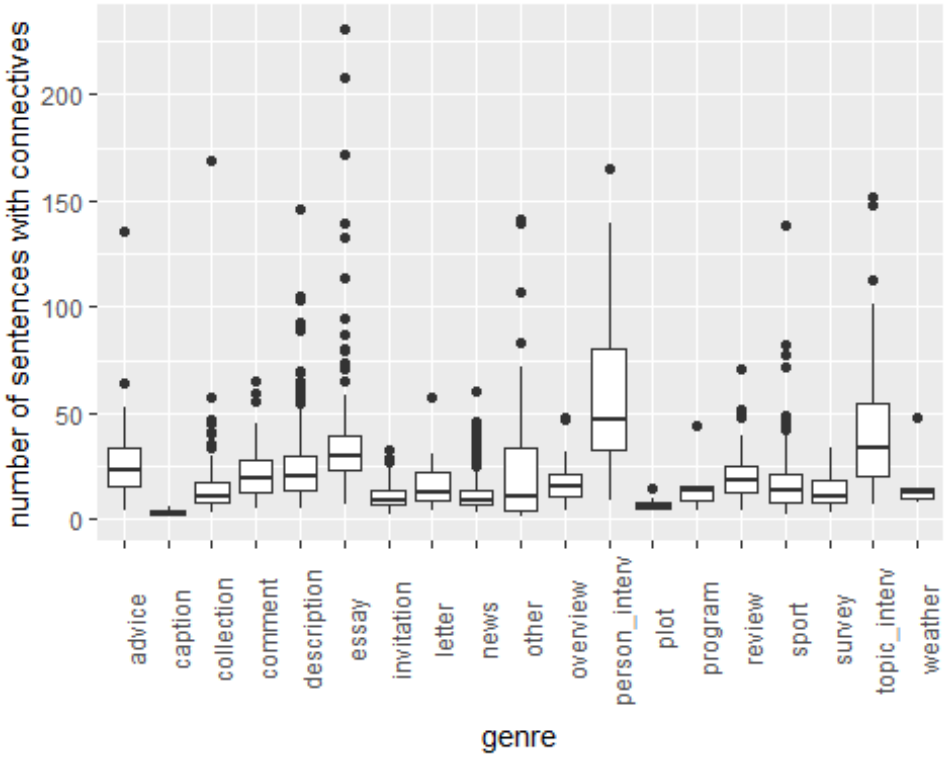


Compare e.g. the tall dark-green column of news in the barplot and the dark-green areas in the columns in this histogram. Together, they tell us: news contribute the largest text bulk to the corpus (barplot) and the texts are mostly short, typically between 10-20 sentences (histogram with color mapping for genres).

How Many How Long Texts Are There in the Corpus for Each Genre? (II)

The colored histogram above suffers from too many similar colors. A row of *boxplots* with whiskers presents the same information textwise and in a more transparent way:

```
ggplot(doclen_set, aes(x = genre, y = number_of_sentences)) +
    geom_boxplot() + theme(axis.text.x = element_text(angle = 90)) +
    ylab("number of sentences with connectives")
```



Boxplot Interpretation: for each genre, all observations sorted from min to max. First quartile: 25% of the observations, second quartile (median): 50% of the observations, third quartile: 75% of the observations, fourth = maximum. Range: max-min. Box: "IQR" (interquartile range): difference between 3rd and 1st quartile. Whiskers: 1,5*IQR. Beyond whiskers: *outliers*.

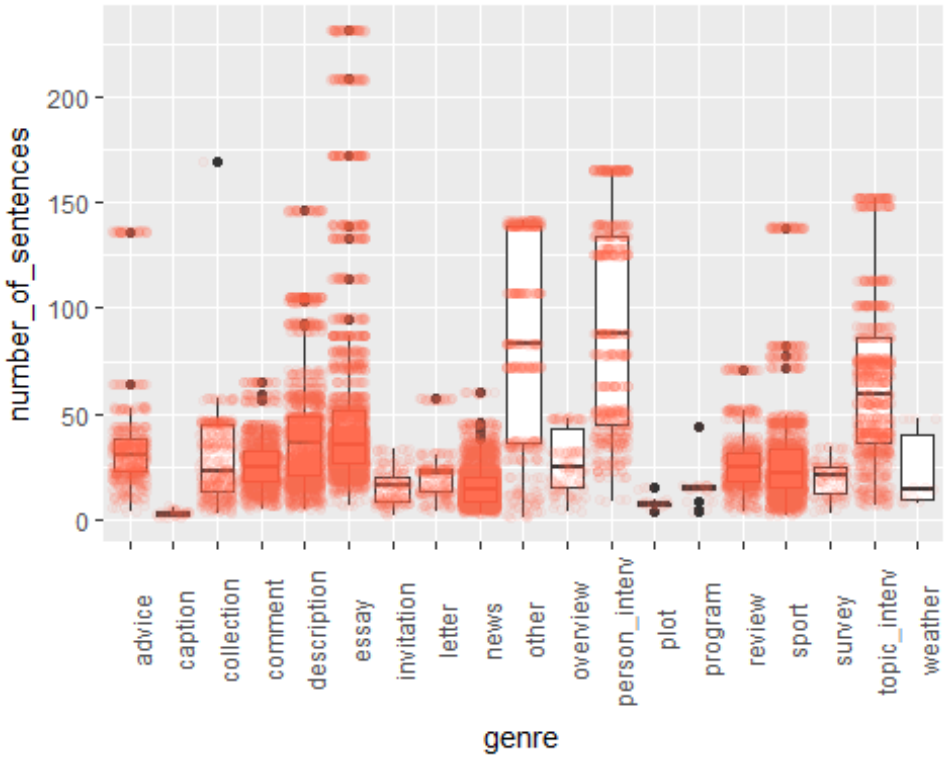
Here: outliers only upwards - quite short texts occur in all genres, but the really long texts in just a few and in even fewer genres they are common! Cf. person_interview and essay.

How Many Connectives Are There in the Different Texts, Genrewise?

When we clutter the boxplots above with a color-mapped additional variable, we will get a first glimpse at how many connectives there are in the individual texts in the individual genres.

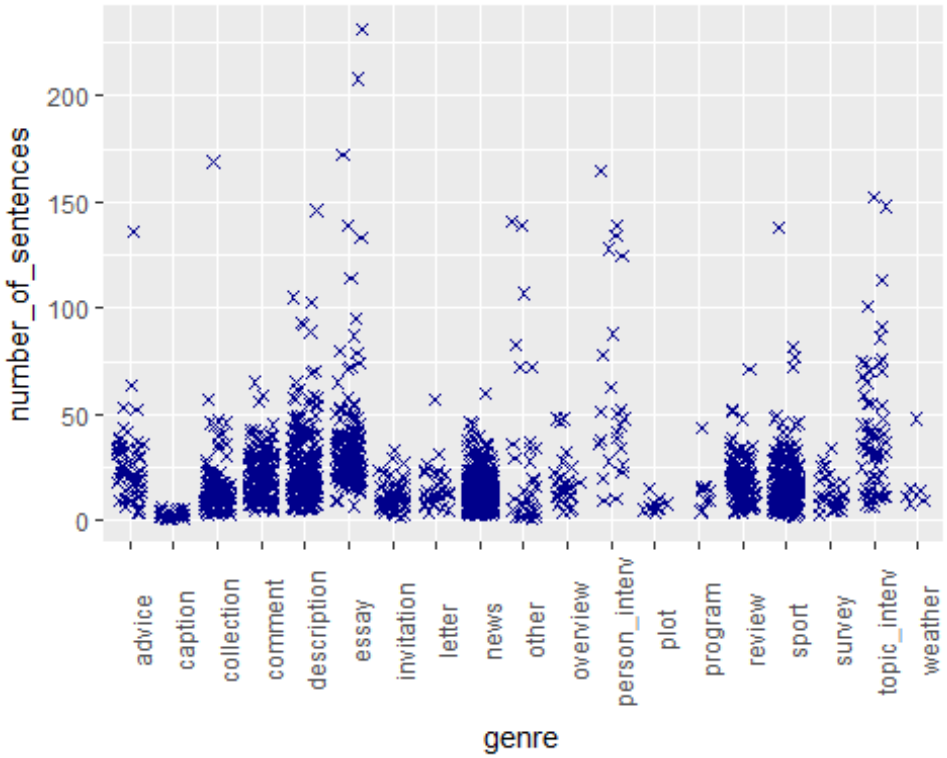
Each tomato-colored point represents one occurrence of a connective in a document of the length given by the y-axis. Since they are so many points that they would hide the boxplots, we made them transparent. With the current plot parameters it takes 500 points over each other to appear as one fully opaque point. The parameter responsible for opacity is called alpha. We will get few opaque points in the plot, since we have also *jittered* them, so that they overlap less frequently.

```
ggplot(pdt30, aes(x = genre, y = number_of_sentences)) + geom_boxplot() +
    geom_jitter(alpha = 5/100, col = "tomato") + theme(axis.text.x = element_
text(angle = 90))
```



NB, the current plot does not represent the individual texts. We may see horizontal tomato-colored clouds and think that they all belong to the same document, but in fact they belong to all documents within the given genre that have the same sentence length. A scatterplot shows the individual text. We will check whether there are many documents with the same length within a single genre.

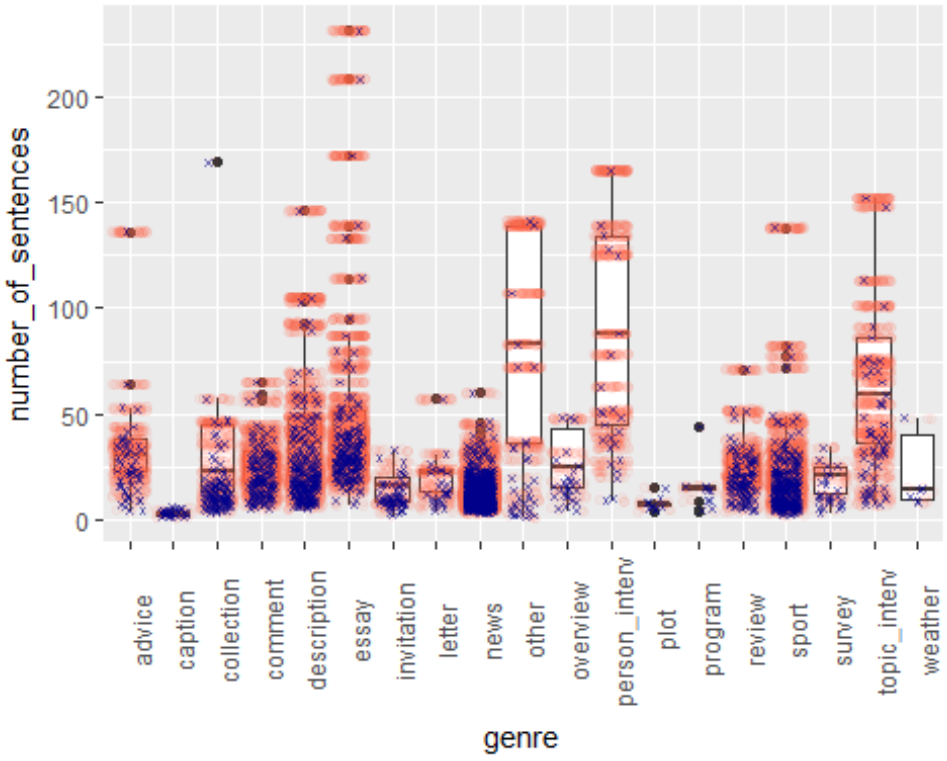
```
ggplot(doclen_set, aes(x = genre, y = number_of_sentences)) +
    geom_point(color = "darkblue", shape = 4, position = position_jitter(heig
ht = 0,
        width = 0.3)) + theme(axis.text.x = element_text(angle = 90))
```



Analysis of Outliers

Outliers in this case are primarily texts with atypically many sentences for the given genre. The length of text itself is not a reason for discarding such a text, but let's look closer at how many connectives (roughly) are associated with each text outlier.

```
indiv_docs <- select(doclen_set, -1)
ggplot(pdt30, aes(x = genre, y = number_of_sentences)) + geom_boxplot() +
    geom_jitter(alpha = 5/100, col = "tomato") + theme(axis.text.x = element_
text(angle = 90)) +
    geom_point(data = indiv_docs, color = "darkblue", shape = 4,
        position = position_jitter(height = 0, width = 0.3),
        alpha = 5/10, size = 2/3)
```



In comparison to the original boxplot with tomato-colored transparent points representing connectives, we combined the boxplot including the individual connectives (transparent tomato points) with the scatterplot above. We got ugly clutter in the homogenous genres characterized with short texts, but on the other hand we have more information about the outliers.

`Collection` outlier really just one connective in such a long text?

```
dplyr::filter(pdt30, genre == "collection", number_of_sentences >
    160)
```

```
## # A tibble: 1 × 6
##    document_id      genre number_of_sentences           sentence_id
```

```
##         <fctr>     <fctr>               <int>                <fctr>
## 1 mf930713_002 collection                 169 t-mf930713-002-p82s2B
## # ... with 2 more variables: discourse_type <fctr>, discourse_class <fctr>
```

```
dplyr::filter(pdt30, document_id == "mf930713_002")
```

```
## # A tibble: 1 × 6
##    document_id      genre number_of_sentences           sentence_id
##         <fctr>     <fctr>               <int>                <fctr>
## 1 mf930713_002 collection                 169 t-mf930713-002-p82s2B
## # ... with 2 more variables: discourse_type <fctr>, discourse_class <fctr>
```

Very odd, discard.

```
pdt30_original <- pdt30  #save the original
pdt30 <- dplyr::filter(filter(pdt30, document_id != "mf930713_002"))
```

Check that it has been removed.

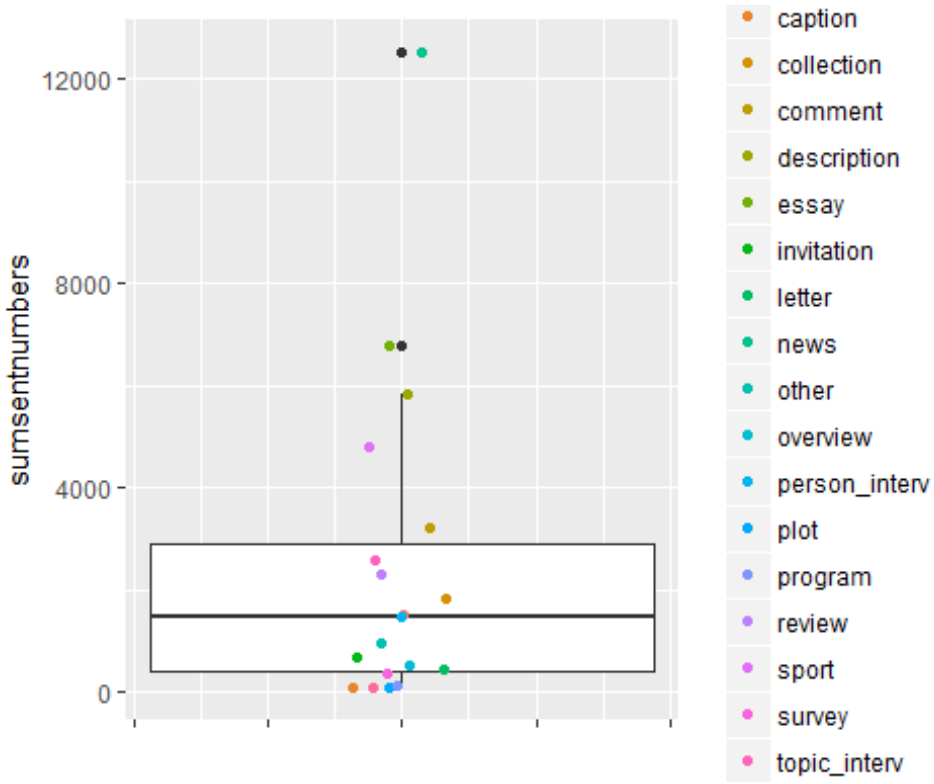
```
dplyr::filter(pdt30, document_id == "mf930713_002")
```

```
## # A tibble: 0 × 6
## # ... with 6 variables: document_id <fctr>, genre <fctr>,
## #   number_of_sentences <int>, sentence_id <fctr>, discourse_type <fctr>,
## #   discourse_class <fctr>
```

A similar one is in `program`. However, it is much shorter. Let's keep it. A bigger issue seems to be the entire genre. It contains very few texts (see barplot - histogram does not say anything about bulk text size!!! Cf. how small the `news` appears in boxplot!!!). We may be forced to discard some genres, if it turns out that they provide too little data. More about this later.

Just for fun - let's make a boxplot of bulk text and color the outliers according to genre. This says a tiny bit more than barplot - we get the comparison with median.

```
ggplot(sentsums, aes(x = 1, y = sumsentnumbers)) + geom_boxplot() +
    theme(axis.text.x = element_blank()) + xlab("") + geom_point(aes(y = sums
entnumbers,
    col = genre), position = position_jitter(height = 0, width = 0.1))
```

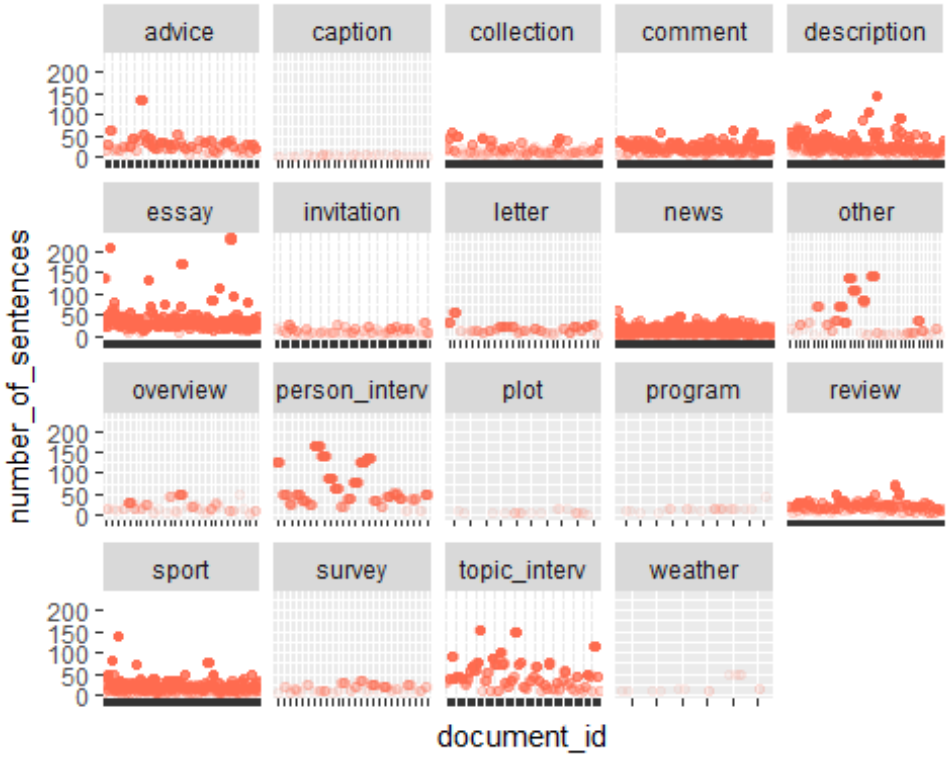


The boxplot says the small text bulks are not really outliers. Let's keep them so far and look at the proportions of connectives within each genre.

How Many Connectives Does Each Genre Contribute?

We will see a faceted scatterplot. Each facet comprises one genre. Each point represents one occurrence of a connective. Besides, their vertical position indicates in how long a document each connective occurred. We see that the fixed y-scale ranges between 0 to 200+ sentences. The x-scale is set individually for each genre. The more ticks on the x-axis, the more documents the given genre comprises.

```
ggplot(pdt30, aes(x = document_id, y = number_of_sentences)) +
    geom_jitter(alpha = 0.06, col = "tomato") + theme(axis.text.x = element_b
lank()) +
    facet_wrap(~genre, scales = "free_x")
```



`Program`, `weather`, and `plot` can contain too little data. `Caption` would, according to the bulk texts size shown by the barplot, be the same case, but this diagram shows that it comprises many more texts. All of them have very small numbers of sentences. Perhaps they have a similar rate of connectives per sentence (we can't tell now!), but on the whole there are many more connectives in `caption` than in the others.

Let's look at the occurrences of individual discourse classes in the genres to find out whether we have enough data in each genre to compare the proportions of discourse classes!

Proportions of Discourse Classes in Genres

There are 4 discourse classes and 19 genres. We want to compare the proportions of the discourse classes across the genres. Therefore we need the numbers of each discourse class per genre.

Let's create a contingency table. It tells us how many times which combination of genre and discourse_class occurred.

```
(cont_tab <- xtabs(formula = ~genre + discourse_class, data = pdt30))
```

```
##                discourse_class
## genre           CONTINGENCY CONTRAST EXPANSION TEMPORAL
##   advice                242      182       293       24
##   caption                 8        9        20        1
##   collection            115      130       225       37
##   comment               483      523       656       69
##   description           520      651      1034      100
##   essay                 913     1092      1569      183
##   invitation             31       52       128        7
##   letter                 90       70       111        8
##   news                  983     1289      1979      259
##   other                 107      116       250       48
##   overview               18       35        78        3
##   person_interv         173      227       306       61
##   plot                    5        5        10        2
##   program                 1        7        14        4
##   review                209      346       482       35
##   sport                 464      794      1095      172
##   survey                 33       36        68        4
##   topic_interv          306      374       518       49
##   weather                 0        0        14        0
```

```
# table(pdt30[,c(2,6)]) #this does the same thing
# round(prop.table(table(pdt30[,c(2,6)])),4)*100
```

Null Hypothesis and Alternative Hypothesis

We have two categorical variables - genre and discourse_class, and we want to find out whether they are dependent from each other. The most common statistical test for this task (dependency and two categorical variables) is *Pearson's Chi-Squared test* (aka *Chisq*). It will test whether the genres differ in their proportions of the discourse classes enough to be such very unlikely by mere chance. The test measures the probability of the differences arising by chance.

What is actually chance in this context? It's basically the chance that having taken a few different annotated texts from each genre would result in yet very different results showing different trends; e.g.: our sample suggests that the genres do not differ so much in expansion (most frequent everywhere) and temporal(least frequent everywhere), but there are differences in proportions of contrast and contingency to each other. If this occurred by

chance, a different sample would suggest, say, that the genres differ most strongly in the proportion of temporal relations to contingency, a yet different sample would suggest that there are no differences between genres at all, etc.

So, the Chi-Squared test tests the probability of chance in the observed results. If this probability is low enough, we can reject the *Null Hypothesis*, which says that there is no dependency between genre and discourse class and all differences we may observe are plain chance. When we can reject the Null Hypothesis, we can claim that we have observed a *statistically significant* dependency between the two variables. However, we may *never* say that we see our hypothesis (aka *Alternative Hypothesis*) verified or confirmed, since there is no way in statistics to do that! All we can do is reject the Null Hypothesis. The probability of chance at which we decide that something does not happen by chance (aka *alpha level*)must be determined *BEFORE* the experiment. The usual alpha level lies at 0.05 p-value, which we will get from the test. (E.g. clinical studies may require an even lower alpha level to make really sure the effect of a tested drug has not occurred by chance.)

One more word of caution - statistical significance does not say anything about the *effect size*. The proportions may differ with almost no probability of chance, so that we could see the difference in any sample we would take, but at the same time, the difference can always be equally small!

As a first approximation, we run the Pearson Chi-Squared test on the contingency table we have made:

```
summary(cont_tab)
```

```
## Call: xtabs(formula = ~genre + discourse_class, data = pdt30)
## Number of cases in table: 20555
## Number of factors: 2
## Test for independence of all factors:
##  Chisq = 297.53, df = 54, p-value = 2.257e-35
##  Chi-squared approximation may be incorrect
```

At first sight, we have got an interesting result, since the p-value is much lower than 0.05. However, we have got a warning that `Chi-squared approximation may be incorrect`. This means that our data has violated some prerequisite of this test and possibly invalidated the entire test result.

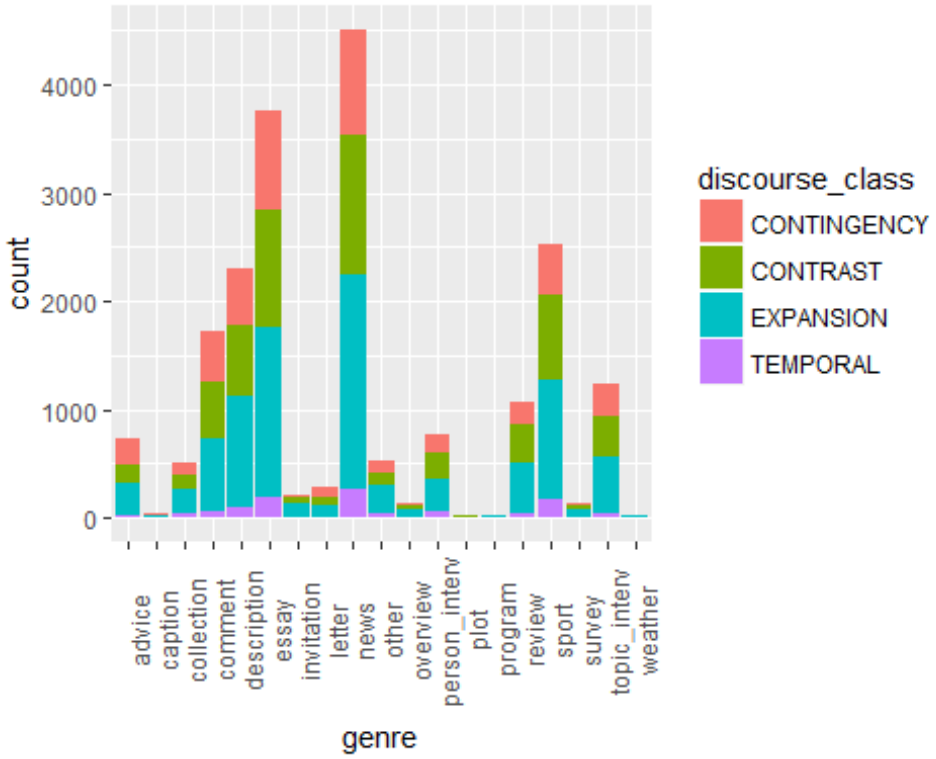
The prerequisites for Chisq are:

1. The data are not correlated (e.g. a before/after scenario)
2. The sample is large enough. Large enough means that each cell having at least 5 *EXPECTED* observations.

The first point is a non-issue in our setup, but let's have a look at the data again how many observations we have in the cells. These are of course just *observed* results, but we will have the chance to identify the most problematic cells at first glance.

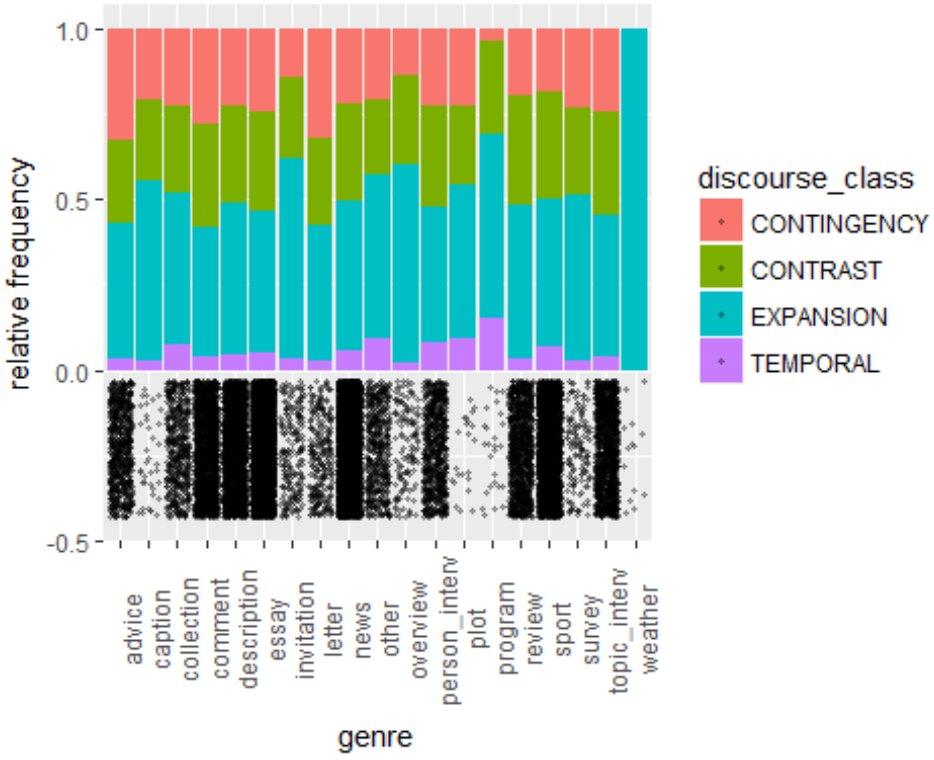
The first diagram shows the genrewise proportions between discourse classes in absolute numbers.

```
ggplot(pdt30, aes(x = genre, fill = discourse_class)) + geom_bar(position = "
stack") +
    theme(axis.text.x = element_text(angle = 90))
```



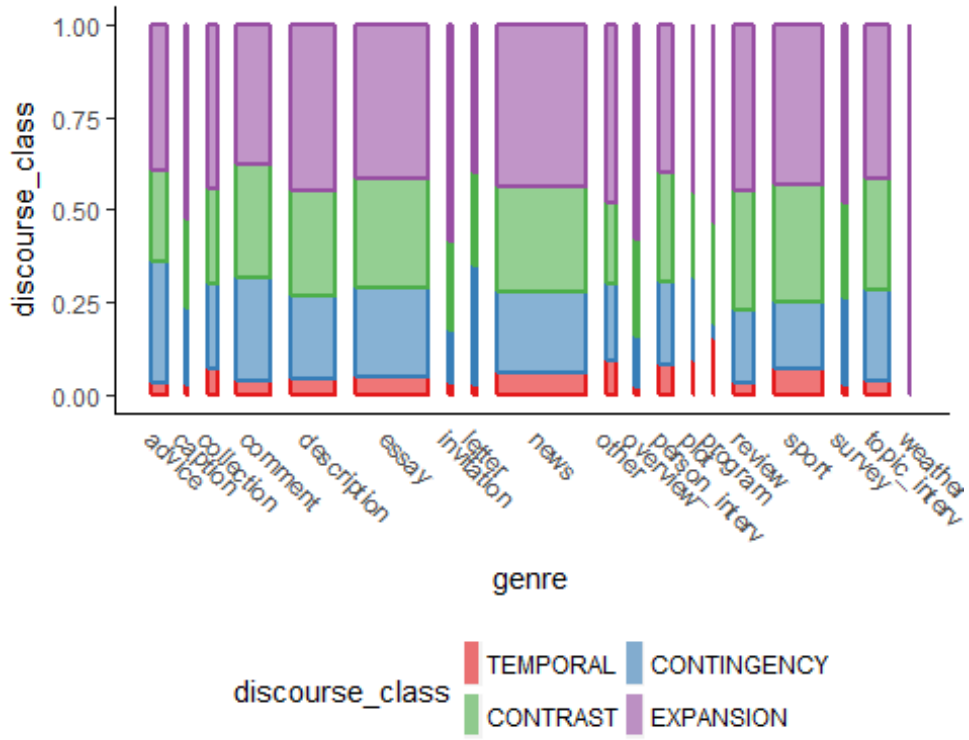
To show the proportions better and preserve the dimension of the data size as much as possible, we transform the barplot with stacked bars to a barplot with filled bars and add a rug with data points jittered and made transparent.

```
ggplot(pdt30, aes(x = genre, fill = discourse_class)) + geom_bar(position = "
fill") +
    theme(axis.text.x = element_text(angle = 90)) + geom_point(aes(y = -0.23)
,
    size = 0.75, alpha = 0.3, position = position_jitter(width = 0.4,
        height = 0.2)) + ylab("relative frequency")
```



We get perhaps the best view using a simple *mosaic plot*. The width of the panels represents their proportion of the total data size.

```
plotluck(data = pdt30, formula = discourse_class ~ genre)
```



Two last views on the genrewise proportions between discourse classes in faceted diagrams. We create a base of the two plots to share and store it in a variable:

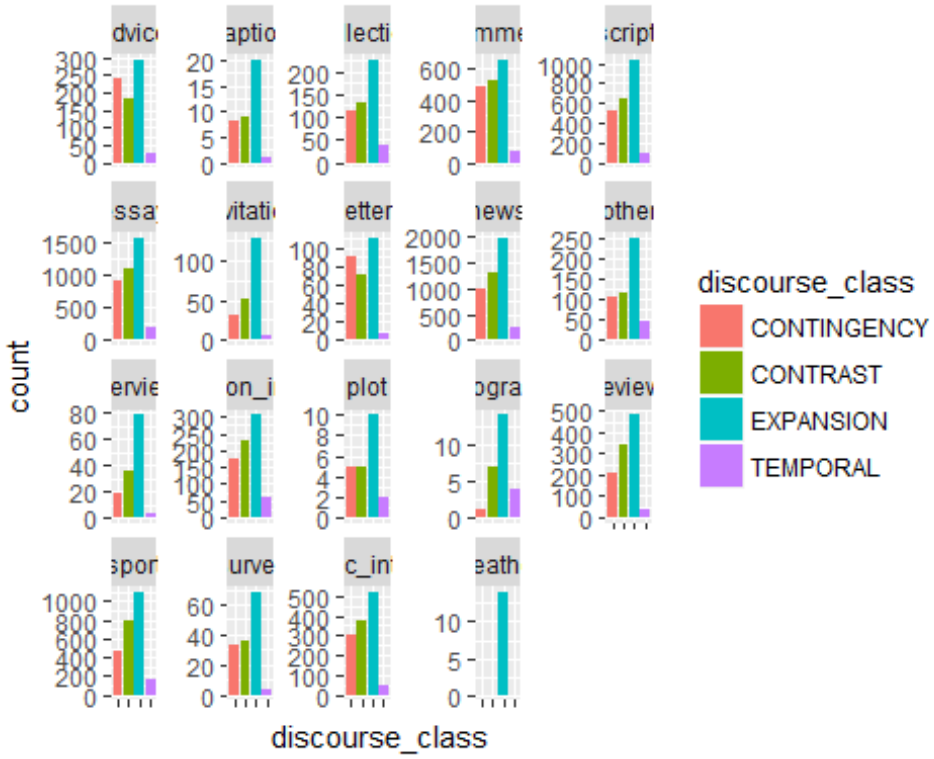
```
mappings_01 <- ggplot(data = pdt30, aes(x = discourse_class,
    fill = discourse_class))
```

A faceted barplot of the four discourse classes with unified y-scale.

```
mappings_01 + geom_bar(position = "dodge") + facet_wrap(~genre,
    scales = "fixed") + theme(axis.text.x = element_blank())
```

The same barplot with individual y-scales - better view of proportions.

```
mappings_01 + geom_bar(position = "dodge") + facet_wrap(~genre,
    scales = "free_y") + theme(axis.text.x = element_blank()) +
    scale_y_continuous(breaks = scales::pretty_breaks())
```



Apparently there is a problem with the `weather` genre lacking three discourse classes of four expected. This one is sure to be discarded or merged with another one, if Chisq is to be used. Other potentially problematic genres are `caption`, `plot`, and `program`.

Let's compute how many EXPECTED observations our data set has to be absolutely sure that the test outcome is reliable.

What are the EXPECTED values? These are values that would be ideally observed if there was no dependency between the variables. The test will compare the expected and the observed values for each cell and estimate the probability with which they differ by chance.

The definition of chisq:

```
chisq = sum((observed - expected)^2/expected)
```

The expected value is computed as
`row_total*column_total/total_number_of_observations`

Computing Expected Observations for Each Cell in a Contingency table

Our contingency table again:

```
cont_tab
```

```
##                discourse_class
## genre           CONTINGENCY CONTRAST EXPANSION TEMPORAL
##   advice                242      182       293       24
##   caption                 8        9        20        1
##   collection            115      130       225       37
##   comment               483      523       656       69
##   description           520      651      1034      100
##   essay                 913     1092      1569      183
##   invitation             31       52       128        7
##   letter                 90       70       111        8
##   news                  983     1289      1979      259
##   other                 107      116       250       48
##   overview               18       35        78        3
##   person_interv         173      227       306       61
##   plot                    5        5        10        2
##   program                 1        7        14        4
##   review                209      346       482       35
##   sport                 464      794      1095      172
##   survey                 33       36        68        4
##   topic_interv          306      374       518       49
##   weather                 0        0        14        0
```

For the sake of further computations, we transform it into a matrix. Only the data structure changes.

```
(cont_matrix <- as.matrix(cont_tab))
```

```
##                discourse_class
## genre           CONTINGENCY CONTRAST EXPANSION TEMPORAL
##   advice                242      182       293       24
##   caption                 8        9        20        1
##   collection            115      130       225       37
##   comment               483      523       656       69
##   description           520      651      1034      100
##   essay                 913     1092      1569      183
##   invitation             31       52       128        7
##   letter                 90       70       111        8
##   news                  983     1289      1979      259
##   other                 107      116       250       48
##   overview               18       35        78        3
##   person_interv         173      227       306       61
##   plot                    5        5        10        2
##   program                 1        7        14        4
##   review                209      346       482       35
##   sport                 464      794      1095      172
```

```
##   survey                33       36        68        4
##   topic_interv         306      374       518       49
##   weather                0        0        14        0
```

The matrix still contains our contingency table. Each cell contains our observed values. The expected value in a cell is computed as `row_total*column_total/total_number_of_observations`. Let's compute the sums for rows and columns as well as the sum of observations:

```
sum(cont_matrix)

## [1] 20555

colSums(cont_matrix)

## CONTINGENCY    CONTRAST   EXPANSION    TEMPORAL
##        4701        5938        8850        1066

rowSums(cont_matrix)

##        advice       caption    collection       comment   description
##           741            38           507          1731          2305
##         essay    invitation        letter          news         other
##          3757           218           279          4510           521
##      overview person_interv          plot       program        review
##           134           767            22            26          1072
##         sport        survey  topic_interv       weather
##          2525           141          1247            14
```

We put these numbers into the formula and compute the expected value for the first row of the first column:

```
row <- 1
column <- 1
colSums(cont_matrix)[column]/sum(cont_matrix) * rowSums(cont_matrix)[row]/sum
(cont_matrix) *
    sum(cont_matrix)

## CONTINGENCY
##    169.4693
```

Let's transform this into a new function called `expected_cell`, with which we can later check random cells after a bulk computation for all cells at once. The function needs the name of the matrix and the indices of the row and column of the cell we ask about.

```
expected_cell <- function(my_matrix, my_row, my_column) {
    my_cell <- colSums(my_matrix)[my_column]/sum(colSums(my_matrix)) *
        rowSums(my_matrix)[my_row]/sum(colSums(my_matrix)) *
        sum(colSums(my_matrix))
    my_cell <- round(my_cell, 1)
    return(my_cell)
}
```

```
expected_cell(my_matrix = cont_matrix, my_row = 1, my_column = 1)
```

```
## CONTINGENCY
##       169.5
```

```
expected_cell(my_matrix = cont_matrix, my_row = 2, my_column = 1)
```

```
## CONTINGENCY
##         8.7
```

Now we can do a bulk computation very fast by calculating with the entire matrix at the same time. We can perform all arithmetic operations on two numerical matrices that we can perform on two numbers. When the matrices have the same dimensions, the given arithmetic operation is performed for each corresponding pair and the outcome is a matrix of the resulting values, which again has the same dimensions as the original matrices.

Let's illustrate it on a small example:

```
(first_matrix <- matrix(c(1, 2, 10, 20, 3, 30), nrow = 2, ncol = 2,
    byrow = FALSE))
```

```
##      [,1] [,2]
## [1,]    1   10
## [2,]    2   20
```

```
(second_matrix <- matrix(c(1, 2, 10, 20, 3, 30), nrow = 2, ncol = 2,
    byrow = TRUE))
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]   10   20
```

```
(third_matrix <- first_matrix + second_matrix)
```

```
##      [,1] [,2]
## [1,]    2   12
## [2,]   12   40
```

The first matrix, first row, first column got added to the second matrix, first row, first column, etc. Note the `byrow` parameter in the `matrix` function. It tells the function how to fill the matrix with the vector we feed it - column-wise (default) or row-wise. The vector was identical in both cases. It was the `byrow` parameter that made the resulting matrices look differently! We are going to use this trick soon.

Back to our contingency table and the expected-value formula

`row_total*column_total/total_number_of_observations`. To bulk-compute all cells at once, we need to multiply two matrices and then to multiply the resulting matrix with the total number of observations.

The first matrix will have the same dimensions as our contingency table. We will fill each column with the sum of values in the given column divided by the number of observations. There are 19 observations (= rows) of 4 variables (= columns). That means we have to repeat each column sum nineteen times. We will get a 76-value vector, which we will break down into a 4 x 19 matrix filled *by column*.

This is the numbers we will repeat 19 times each:

```
colSums(cont_matrix)/sum(cont_matrix)
```

```
## CONTINGENCY    CONTRAST   EXPANSION    TEMPORAL
##  0.22870348  0.28888348  0.43055218  0.05186086
```

Here is how we create the vector:

```
rep(colSums(cont_matrix)/sum(cont_matrix), each = 19)
```

```
## CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY
##  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348
## CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY
##  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348
## CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY CONTINGENCY
##  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348  0.22870348
## CONTINGENCY    CONTRAST    CONTRAST    CONTRAST    CONTRAST    CONTRAST
##  0.22870348  0.28888348  0.28888348  0.28888348  0.28888348  0.28888348
##    CONTRAST    CONTRAST    CONTRAST    CONTRAST    CONTRAST    CONTRAST
##  0.28888348  0.28888348  0.28888348  0.28888348  0.28888348  0.28888348
##    CONTRAST    CONTRAST    CONTRAST    CONTRAST    CONTRAST    CONTRAST
##  0.28888348  0.28888348  0.28888348  0.28888348  0.28888348  0.28888348
##    CONTRAST    CONTRAST   EXPANSION   EXPANSION   EXPANSION   EXPANSION
##  0.28888348  0.28888348  0.43055218  0.43055218  0.43055218  0.43055218
##   EXPANSION   EXPANSION   EXPANSION   EXPANSION   EXPANSION   EXPANSION
##  0.43055218  0.43055218  0.43055218  0.43055218  0.43055218  0.43055218
##   EXPANSION   EXPANSION   EXPANSION   EXPANSION   EXPANSION   EXPANSION
##  0.43055218  0.43055218  0.43055218  0.43055218  0.43055218  0.43055218
##   EXPANSION   EXPANSION   EXPANSION    TEMPORAL    TEMPORAL    TEMPORAL
##  0.43055218  0.43055218  0.43055218  0.05186086  0.05186086  0.05186086
##    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL
##  0.05186086  0.05186086  0.05186086  0.05186086  0.05186086  0.05186086
##    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL
##  0.05186086  0.05186086  0.05186086  0.05186086  0.05186086  0.05186086
##    TEMPORAL    TEMPORAL    TEMPORAL    TEMPORAL
##  0.05186086  0.05186086  0.05186086  0.05186086
```

Here comes the matrix of these column sums:

```
(mat_cols <- rep(colSums(cont_matrix)/sum(cont_matrix), each = nrow(cont_matr
ix)) %>%
    matrix(nrow = nrow(cont_matrix), ncol = ncol(cont_matrix)))
```

```
##            [,1]      [,2]      [,3]       [,4]
##  [1,] 0.2287035 0.2888835 0.4305522 0.05186086
```

```
##  [2,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [3,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [4,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [5,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [6,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [7,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [8,] 0.2287035 0.2888835 0.4305522 0.05186086
##  [9,] 0.2287035 0.2888835 0.4305522 0.05186086
## [10,] 0.2287035 0.2888835 0.4305522 0.05186086
## [11,] 0.2287035 0.2888835 0.4305522 0.05186086
## [12,] 0.2287035 0.2888835 0.4305522 0.05186086
## [13,] 0.2287035 0.2888835 0.4305522 0.05186086
## [14,] 0.2287035 0.2888835 0.4305522 0.05186086
## [15,] 0.2287035 0.2888835 0.4305522 0.05186086
## [16,] 0.2287035 0.2888835 0.4305522 0.05186086
## [17,] 0.2287035 0.2888835 0.4305522 0.05186086
## [18,] 0.2287035 0.2888835 0.4305522 0.05186086
## [19,] 0.2287035 0.2888835 0.4305522 0.05186086
```

Now the matrix of the row sums. Mind that the vector will have to be broken into a matrix *by rows*!

```
(mat_rows <- rep(rowSums(cont_matrix)/sum(cont_matrix), each = ncol(cont_matr
ix)) %>%
    matrix(nrow = nrow(cont_matrix), ncol = ncol(cont_matrix),
        byrow = TRUE))
```

```
##               [,1]         [,2]         [,3]         [,4]
##  [1,] 0.0360496230 0.0360496230 0.0360496230 0.0360496230
##  [2,] 0.0018486986 0.0018486986 0.0018486986 0.0018486986
##  [3,] 0.0246655315 0.0246655315 0.0246655315 0.0246655315
##  [4,] 0.0842130868 0.0842130868 0.0842130868 0.0842130868
##  [5,] 0.1121381659 0.1121381659 0.1121381659 0.1121381659
##  [6,] 0.1827779129 0.1827779129 0.1827779129 0.1827779129
##  [7,] 0.0106056920 0.0106056920 0.0106056920 0.0106056920
##  [8,] 0.0135733398 0.0135733398 0.0135733398 0.0135733398
##  [9,] 0.2194113354 0.2194113354 0.2194113354 0.2194113354
## [10,] 0.0253466310 0.0253466310 0.0253466310 0.0253466310
## [11,] 0.0065190951 0.0065190951 0.0065190951 0.0065190951
## [12,] 0.0373145220 0.0373145220 0.0373145220 0.0373145220
## [13,] 0.0010702992 0.0010702992 0.0010702992 0.0010702992
## [14,] 0.0012648991 0.0012648991 0.0012648991 0.0012648991
## [15,] 0.0521527609 0.0521527609 0.0521527609 0.0521527609
## [16,] 0.1228411579 0.1228411579 0.1228411579 0.1228411579
## [17,] 0.0068596449 0.0068596449 0.0068596449 0.0068596449
## [18,] 0.0606665045 0.0606665045 0.0606665045 0.0606665045
## [19,] 0.0006810995 0.0006810995 0.0006810995 0.0006810995
```

The first matrix (`mat_cols`) multiplied with the second matrix (`mat_rows`) multiplied by the total of all observations will give us the matrix of expected values. We will round the result to one decimal place at the same time.

```
(exp_matrix <- (mat_cols * mat_rows * sum(cont_matrix)) %>% round(1))
```

```
##         [,1]   [,2]   [,3]  [,4]
##  [1,]  169.5  214.1  319.0  38.4
##  [2,]    8.7   11.0   16.4   2.0
##  [3,]  116.0  146.5  218.3  26.3
##  [4,]  395.9  500.1  745.3  89.8
##  [5,]  527.2  665.9  992.4 119.5
##  [6,]  859.2 1085.3 1617.6 194.8
##  [7,]   49.9   63.0   93.9  11.3
##  [8,]   63.8   80.6  120.1  14.5
##  [9,] 1031.5 1302.9 1941.8 233.9
## [10,]  119.2  150.5  224.3  27.0
## [11,]   30.6   38.7   57.7   6.9
## [12,]  175.4  221.6  330.2  39.8
## [13,]    5.0    6.4    9.5   1.1
## [14,]    5.9    7.5   11.2   1.3
## [15,]  245.2  309.7  461.6  55.6
## [16,]  577.5  729.4 1087.1 130.9
## [17,]   32.2   40.7   60.7   7.3
## [18,]  285.2  360.2  536.9  64.7
## [19,]    3.2    4.0    6.0   0.7
```

We will give the matrix back the original row and column names for easier indexing.

```
row.names(exp_matrix) <- row.names(cont_matrix)
colnames(exp_matrix) <- colnames(cont_matrix)
exp_matrix
```

```
##                CONTINGENCY CONTRAST EXPANSION TEMPORAL
## advice               169.5    214.1     319.0     38.4
## caption                8.7     11.0      16.4      2.0
## collection           116.0    146.5     218.3     26.3
## comment              395.9    500.1     745.3     89.8
## description          527.2    665.9     992.4    119.5
## essay                859.2   1085.3    1617.6    194.8
## invitation            49.9     63.0      93.9     11.3
## letter                63.8     80.6     120.1     14.5
## news                1031.5   1302.9    1941.8    233.9
## other                119.2    150.5     224.3     27.0
## overview              30.6     38.7      57.7      6.9
## person_interv        175.4    221.6     330.2     39.8
## plot                   5.0      6.4       9.5      1.1
## program                5.9      7.5      11.2      1.3
## review               245.2    309.7     461.6     55.6
## sport                577.5    729.4    1087.1    130.9
## survey                32.2     40.7      60.7      7.3
```

```
## topic_interv        285.2    360.2     536.9     64.7
## weather               3.2      4.0       6.0      0.7
```

Let's perform a few random checks with our one-cell function `expected_cell`:

```
expected_cell(my_matrix = exp_matrix, my_row = 1, my_column = 4)

## TEMPORAL
##     38.4

expected_cell(my_matrix = exp_matrix, my_row = 19, my_column = 1)

## CONTINGENCY
##         3.2

expected_cell(my_matrix = exp_matrix, my_row = 3, my_column = 2)

## CONTRAST
##    146.5
```

The results of cell-wise vs. bulk-wise computations agree. Our matrix of expected values is correct.

Finally, we will identify the *names* of rows (i.e. genres), in which the expected number of observations is less than 5.

```
which(exp_matrix < 5, arr.ind = TRUE)

##         row col
## weather  19   1
## weather  19   2
## caption   2   4
## plot     13   4
## program  14   4
## weather  19   4
```

This shows that, to be able to use the chisq test, we have to remove `weather`, `caption`, `plot`, and `program`, alternatively merge them and test the expected values again. However, let's assume that an umbrella genre does not make sense. We will rather merge all these with `other`. This means that, in our `pdt30` data set, all genre values equal to these will be replaced with `other`

```
levels(pdt30$genre)[levels(pdt30$genre) %in% c("weather", "caption",
    "plot", "program")] <- "other"
```

Check

```
sum(pdt30$genre %in% c("weather", "caption", "plot", "program"))

## [1] 0

head(pdt30_original[which(pdt30_original$genre %in% c("weather",
    "caption", "plot", "program")), ])
```

```
## # A tibble: 6 × 6
##    document_id   genre number_of_sentences         sentence_id
##          <fctr>  <fctr>               <int>              <fctr>
## 1 cmpr9410_051 caption                   2 t-cmpr9410-051-p3s2
## 2 cmpr9415_055 caption                   2 t-cmpr9415-055-p3s1
## 3 cmpr9415_056 caption                   2 t-cmpr9415-056-p3s1
## 4   ln94200_10    plot                  10   t-ln94200-10-p3s2
## 5   ln94200_11    plot                   6   t-ln94200-11-p2s2
## 6   ln94200_12 program                   9   t-ln94200-12-p2s5
## # ... with 2 more variables: discourse_type <fctr>, discourse_class <fctr>
```

```
head(pdt30[which(pdt30_original$genre %in% c("weather", "caption",
    "plot", "program")), ])
```

```
## # A tibble: 6 × 6
##    document_id  genre number_of_sentences         sentence_id
##          <fctr> <fctr>               <int>              <fctr>
## 1 cmpr9410_051  other                   2 t-cmpr9410-051-p3s2
## 2 cmpr9415_055  other                   2 t-cmpr9415-055-p3s1
## 3 cmpr9415_056  other                   2 t-cmpr9415-056-p3s1
## 4   ln94200_10  other                  10   t-ln94200-10-p3s2
## 5   ln94200_11  other                   6   t-ln94200-11-p2s2
## 6   ln94200_12  other                   9   t-ln94200-12-p2s5
## # ... with 2 more variables: discourse_type <fctr>, discourse_class <fctr>
```

Now we can run the chisq test.

```
(result <- chisq.test(pdt30$genre, pdt30$discourse_class))
```

```
##
##  Pearson's Chi-squared test
##
## data:  pdt30$genre and pdt30$discourse_class
## X-squared = 273.34, df = 42, p-value < 2.2e-16
```

The p-value is much less than 0.05, so we can reject the null hypothesis that denies any association between genre and distribution of discourse classes. No warning occurs, so we can trust it. Let's look at the results closer, though, to see which discourse classes and genres have contributed the most and the least and how to this result. We will be able to see which genres and discourse classes attract/repel each other. All this information is hidden in the object that is returned by the `chisq.test` function. It is a list.

```
str(result)
```

```
## List of 9
##  $ statistic: Named num 273
##   ..- attr(*, "names")= chr "X-squared"
##  $ parameter: Named int 42
##   ..- attr(*, "names")= chr "df"
##  $ p.value  : num 1.1e-35
##  $ method   : chr "Pearson's Chi-squared test"
```

```
##  $ data.name: chr "pdt30$genre and pdt30$discourse_class"
##  $ observed : 'table' int [1:15, 1:4] 242 121 115 483 520 913 31 90 983 18
...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ pdt30$genre          : chr [1:15] "advice" "other" "collection" "
comment" ...
##   .. ..$ pdt30$discourse_class: chr [1:4] "CONTINGENCY" "CONTRAST" "EXPANS
ION" "TEMPORAL"
##  $ expected : num [1:15, 1:4] 169 142 116 396 527 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ pdt30$genre          : chr [1:15] "advice" "other" "collection" "
comment" ...
##   .. ..$ pdt30$discourse_class: chr [1:4] "CONTINGENCY" "CONTRAST" "EXPANS
ION" "TEMPORAL"
##  $ residuals: table [1:15, 1:4] 5.5716 -1.7642 -0.0885 4.3783 -0.3119 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ pdt30$genre          : chr [1:15] "advice" "other" "collection" "
comment" ...
##   .. ..$ pdt30$discourse_class: chr [1:4] "CONTINGENCY" "CONTRAST" "EXPANS
ION" "TEMPORAL"
##  $ stdres   : table [1:15, 1:4] 6.462 -2.04 -0.102 5.21 -0.377 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ pdt30$genre          : chr [1:15] "advice" "other" "collection" "
comment" ...
##   .. ..$ pdt30$discourse_class: chr [1:4] "CONTINGENCY" "CONTRAST" "EXPANS
ION" "TEMPORAL"
##  - attr(*, "class")= chr "htest"
```

The elements `observed` and `expected` contain the contingency tables of observed and expected values, respectively. The expected values are exactly those we have computed before (except for the genres we have merged). Unfortunately, `chisq.test` only computes the expected values when it is able to proceed correctly, which is why we had to find out and remove the problematic genres manually.

```
round(result$expected, 1)
```

```
##                pdt30$discourse_class
## pdt30$genre     CONTINGENCY CONTRAST EXPANSION TEMPORAL
##   advice              169.5    214.1     319.0     38.4
##   other               142.0    179.4     267.4     32.2
##   collection          116.0    146.5     218.3     26.3
##   comment             395.9    500.1     745.3     89.8
##   description         527.2    665.9     992.4    119.5
##   essay               859.2   1085.3    1617.6    194.8
##   invitation           49.9     63.0      93.9     11.3
##   letter               63.8     80.6     120.1     14.5
##   news               1031.5   1302.9    1941.8    233.9
##   overview             30.6     38.7      57.7      6.9
##   person_interv       175.4    221.6     330.2     39.8
##   review              245.2    309.7     461.6     55.6
```

```
##   sport               577.5    729.4    1087.1    130.9
##   survey               32.2     40.7      60.7      7.3
##   topic_interv        285.2    360.2     536.9     64.7
```

```
expected_cell(cont_matrix, 1, 1)  #cf. matrix before genre merging
```

```
## CONTINGENCY
##       169.5
```

Standardized residuals (`stdres`) will tell more about the effect of different genre vs. discourse class combinations. They express the difference between the expected and the observed values (observed minus expected, the whole divided by the square root of expected). Standardized residuals give a better comparison than residuals (`residuals`), because they abstract from the fact that with the same difference x, a large number - x is always larger than a small number - x.

```
result$stdres
```

```
##                pdt30$discourse_class
## pdt30$genre     CONTINGENCY   CONTRAST  EXPANSION   TEMPORAL
##   advice          6.4615765 -2.6468625 -1.9676630 -2.4346740
##   other          -2.0398666 -3.8116713  3.3434255  4.1887851
##   collection     -0.1020027 -1.6335066  0.6094034  2.1712639
##   comment         5.2095058  1.2713551 -4.5289432 -2.3526586
##   description    -0.3769206 -0.7255357  1.8561264 -1.9478011
##   essay           2.3100952  0.2653760 -1.7707918 -0.9637207
##   invitation     -3.0571787 -1.6490094  4.6946794 -1.3221224
##   letter          3.7590890 -1.4095414 -1.1107431 -1.7585667
##   news           -1.9443364 -0.5155531  1.2665335  1.9083080
##   overview       -2.6096549 -0.7095047  3.5542873 -1.5436120
##   person_interv  -0.2116568  0.4405930 -1.8010955  3.5221289
##   review         -2.7016956  2.5136831  1.2955250 -2.9136409
##   sport          -5.7410089  3.0270799  0.3371150  3.9337022
##   survey          0.1514688 -0.8823696  1.2445159 -1.2623203
##   topic_interv    1.4474858  0.8871888 -1.1151822 -2.0648269
```

The values with negative sign mean a repelling effect, the positive ones an attracting effect.

Chisq Warnings

- *NEVER* compute chisq on percentages!!! The contingency table has to contain absolute frequencies.
- *Chisq does not give you the effect size*. What you see in the residuals gives you a hint about which categories attract or repel each other and you can compare the numbers as strengths of these partial trends, but you cannot get a total effect size by adding them together or whatever you might think about!!!

Alternative to Chisq: Fisher's exact test

What about if you cannot use chisq but really need to test your two categorical variables on dependence? Fisher exact test. It is implemented in R, but it is not designed for as many categories as we have. Strictly speaking, it is designed for 2x2 tables only, while we have a 19 x 4 table.

It will certainly throw an error:

```
fisher.test(pdt30$genre, pdt30$discourse_class)
```

```
Error in fisher.test(pdt30\$genre, pdt30\$discourse_class) :
FEXACT error 5.
The hash table key cannot be computed because the largest key is larger than
the largest representable int.
The algorithm cannot proceed.Reduce the workspace size or use another algorit
hm.
```

So if we really really need this test, we have to divide the task into several smaller tables. Let's try and compute whether there is a significant difference in the distribution of discourse classes between advice and invitation:

```
adv_invit <- dplyr::filter(pdt30, genre %in% c("advice", "invitation"))
dplyr::glimpse(adv_invit)
```

```
## Observations: 959
## Variables: 6
## $ document_id         <fctr> cmpr9410_003, cmpr9410_003, cmpr9410_003,...
## $ genre               <fctr> advice, advice, advice, advice, advice, a...
## $ number_of_sentences <int> 16, 16, 16, 16, 16, 29, 29, 29, 29, 29, 29...
## $ sentence_id         <fctr> t-cmpr9410-003-p5s2, t-cmpr9410-003-p5s4,...
## $ discourse_type      <fctr> conj, conj, cond, reason, reason, restr, ...
## $ discourse_class     <fctr> EXPANSION, EXPANSION, CONTINGENCY, CONTIN...
```

```
fisher.test(adv_invit$genre, adv_invit$discourse_class, alternative = "two.si
ded")
```

```
 Error in fisher.test(adv_invit$genre,
 adv_invit$discourse_class, alternative = "two.sided") :
 Bug in FEXACT: gave negative key
```

A bit of trying reveals that at the moment of writing this code (probably conditioned by the given combination computing power, OS, R version etc.), Fisher's test could pass with three of the four labels (no matter which they were) and two genres:

```
adv_invit_contingency <- dplyr::filter(adv_invit, discourse_class %in%
    c("CONTINGENCY", "TEMPORAL", "CONTRAST")) %>% select(one_of("genre",
    "discourse_class"))
dplyr::glimpse(adv_invit_contingency)
```

```
## Observations: 538
## Variables: 2
```

```
## $ genre           <fctr> advice, advice, advice, advice, advice, advic...
## $ discourse_class <fctr> CONTINGENCY, CONTINGENCY, CONTINGENCY, CONTRA...

result <- fisher.test(adv_invit_contingency$genre, adv_invit_contingency$disc
ourse_class)
result

##
##  Fisher's Exact Test for Count Data
##
## data:  adv_invit_contingency$genre and adv_invit_contingency$discourse_cla
ss
## p-value = 0.002287
## alternative hypothesis: two.sided
```

Testing all combinations of genres and labels would require a dedicated script, but calling the test function itself is easy. Fisher test has one more advantage over chisq: you can determine whether your alternative hypothesis is two-sided or one-sided in eiter direction.

If really performed, we would probably have to create a special 2x2 table for each category, e.g. EXPANSION - NOT EXPANSION. If we choose two existing categories instead (e.g. EXPANSION - TEMPORAL), we are discarding data and biasing the results.

One-sided vs. Two-Sided Alternative Hypothesis

Two-sided - Less specific. You simply claim: "The two things are different." One-sided - More specific. You claim: "One thing is more/less ... than the other." When you formulate your hypothesis as one-sided, you need less evidence for the p-value to be low, i.e. result significant, than if you formulated your hypothesis as one-sided. However, you should decide BEFORE you compute it. Reformulating your hypothesis from two-sided to one-sided after you have obtained non-significant results from your two-sided hypothesis equals cheating!!! Also, when you formulate your hypothesis as one-sided just as a precaution, you run the risk of missing the effect on the other side. Imagine you would test a drug to increase the size of orchard apples and really needed to sell it, so you would test it with the one-sided alternative hypothesis "apple trees treated with my drug bear bigger apples", which would prove significant, but in fact, the trees treated with your drug would also bear significantly more really small apples at the same time!