# Chapter 4 Time series features

Ankit Gupta

27/12/2023

The feasts package includes functions for computing FEatures And Statistics from Time Series (hence the name). We have already seen some time series features. For example, the autocorrelations discussed in Section 2.8 can be considered features of a time series — they are numerical summaries computed from the series. Another feature we saw in the last chapter was the Guerrero estimate of the Box-Cox transformation parameter — again, this is a number computed from a time series.

We can compute many different features on many different time series, and use them to explore the properties of the series.

## 4.1 Some simple statistics

Any numerical summary computed from a time series is a feature of that time series — the mean, minimum or maximum, for example. These can be computed using the features() function. For example, let's compute the means of all the series in the Australian tourism data.

```
library(fpp3)
```

```
## -- Attaching packages --------------------------------------------- fpp3 0.5 --

## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.3     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.1
## v lubridate   1.9.3     v fable       0.3.3
## v ggplot2     3.4.4     v fabletools  0.3.4

## -- Conflicts ------------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```
tourism |>
  features(Trips, list(mean = mean)) |>
  arrange(mean)
```

```
## # A tibble: 304 x 4
##    Region          State               Purpose   mean
##    <chr>           <chr>               <chr>    <dbl>
##  1 Kangaroo Island South Australia     Other    0.340
##  2 MacDonnell      Northern Territory  Other    0.449
##  3 Wilderness West Tasmania            Other    0.478
##  4 Barkly          Northern Territory  Other    0.632
##  5 Clare Valley    South Australia     Other    0.898
##  6 Barossa         South Australia     Other    1.02
##  7 Kakadu Arnhem   Northern Territory  Other    1.04
##  8 Lasseter        Northern Territory  Other    1.14
##  9 Wimmera         Victoria            Other    1.15
## 10 MacDonnell      Northern Territory  Visiting 1.18
## # i 294 more rows
```

Here we see that the series with least average number of visits was "Other" visits to Kangaroo Island in South Australia.

Rather than compute one feature at a time, it is convenient to compute many features at once. A common short summary of a data set is to compute five summary statistics: the minimum, first quartile, median, third quartile and maximum. These divide the data into four equal-size sections, each containing 25% of the data. The quantile() function can be used to compute them.

```
tourism |> features(Trips, quantile)
```

```
## # A tibble: 304 x 8
##    Region         State             Purpose    `0%`  `25%`   `50%`  `75%` `100%`
##    <chr>          <chr>             <chr>     <dbl>  <dbl>   <dbl>  <dbl>  <dbl>
##  1 Adelaide       South Australia   Busine~  68.7   134.    153.    177.   242.
##  2 Adelaide       South Australia   Holiday 108.    135.    154.    172.   224.
##  3 Adelaide       South Australia   Other    25.9    43.9   53.8    62.5  107.
##  4 Adelaide       South Australia   Visiti~ 137.    179.    206.    229.   270.
##  5 Adelaide Hills South Australia   Busine~   0       0      1.26    3.92  28.6
##  6 Adelaide Hills South Australia   Holiday   0       5.77   8.52   14.1   35.8
##  7 Adelaide Hills South Australia   Other     0       0      0.908   2.09   8.95
##  8 Adelaide Hills South Australia   Visiti~   0.778   8.91  12.2    16.8   81.1
##  9 Alice Springs  Northern Territo~ Busine~   1.01    9.13  13.3    18.5   34.1
## 10 Alice Springs  Northern Territo~ Holiday   2.81   16.9   31.5    44.8   76.5
## # i 294 more rows
```

Here the minimum is labelled 0% and the maximum is labelled 100%.

All the autocorrelations of a series can be considered features of that series. We can also summarise the autocorrelations to produce new features; for example, the sum of the first ten squared autocorrelation coefficients is a useful summary of how much autocorrelation there is in a series, regardless of lag.

We can also compute autocorrelations of the changes in the series between periods. That is, we "difference" the data and create a new time series consisting of the differences between consecutive observations. Then we can compute the autocorrelations of this new differenced series. Occasionally it is useful to apply the same differencing operation again, so we compute the differences of the differences. The autocorrelations of this double differenced series may provide useful information.

Another related approach is to compute seasonal differences of a series. If we had monthly data, for example, we would compute the difference between consecutive Januaries, consecutive Februaries, and so on. This

enables us to look at how the series is changing between years, rather than between months. Again, the autocorrelations of the seasonally differenced series may provide useful information.

The feat_acf() function computes a selection of the autocorrelations discussed here. It will return six or seven features:

- the first autocorrelation coefficient from the original data;
- the sum of squares of the first ten autocorrelation coefficients from the original data;
- the first autocorrelation coefficient from the differenced data;
- the sum of squares of the first ten autocorrelation coefficients from the differenced data;
- the first autocorrelation coefficient from the twice differenced data;
- the sum of squares of the first ten autocorrelation coefficients from the twice differenced data;
- For seasonal data, the autocorrelation coefficient at the first seasonal lag is also returned.

When applied to the Australian tourism data, we get the following output.

```
tourism |> features(Trips, feat_acf)
```

```
## # A tibble: 304 x 10
##    Region         State Purpose     acf1 acf10 diff1_acf1 diff1_acf10 diff2_acf1
##    <chr>          <chr> <chr>      <dbl> <dbl>      <dbl>       <dbl>      <dbl>
##  1 Adelaide       Sout~ Busine~  0.0333  0.131     -0.520       0.463     -0.676
##  2 Adelaide       Sout~ Holiday  0.0456  0.372     -0.343       0.614     -0.487
##  3 Adelaide       Sout~ Other    0.517   1.15      -0.409       0.383     -0.675
##  4 Adelaide       Sout~ Visiti~  0.0684  0.294     -0.394       0.452     -0.518
##  5 Adelaide Hills Sout~ Busine~  0.0709  0.134     -0.580       0.415     -0.750
##  6 Adelaide Hills Sout~ Holiday  0.131   0.313     -0.536       0.500     -0.716
##  7 Adelaide Hills Sout~ Other    0.261   0.330     -0.253       0.317     -0.457
##  8 Adelaide Hills Sout~ Visiti~  0.139   0.117     -0.472       0.239     -0.626
##  9 Alice Springs  Nort~ Busine~  0.217   0.367     -0.500       0.381     -0.658
## 10 Alice Springs  Nort~ Holiday -0.00660 2.11      -0.153       2.11      -0.274
## # i 294 more rows
## # i 2 more variables: diff2_acf10 <dbl>, season_acf1 <dbl>
```

## 4.3 STL Features

The STL decomposition discussed in Chapter 3 is the basis for several more features.

A time series decomposition can be used to measure the strength of trend and seasonality in a time series. Recall that the decomposition is written as $y_t = T_t + S_t + R_t$, where $T_t$ is the smoothed trend component, $S_t$ is the seasonal component and $R_t$ is a remainder component. For strongly trended data, the seasonally adjusted data should have much more variation than the remainder component. Therefore $\frac{Var(R_t)}{Var(T_t+R_t)}$ should be relatively small. But for data with little or no trend, the two variances should be approximately the same. So we define the strength of trend as:

$$F_T = \max(0, 1 - \frac{Var(R_t)}{Var(T_t + R_t)})$$

This will give a measure of the strength of the trend between 0 and 1. Because the variance of the remainder might occasionally be even larger than the variance of the seasonally adjusted data, we set the minimal possible value of $F_T$ equal to zero.

The strength of seasonality is defined similarly, but with respect to the detrended data rather than the seasonally adjusted data:

$$F_S = \max(0, 1 - \frac{Var(R_t)}{Var(S_t + R_t)})$$

A series with seasonal strength $F_S$ close to 0 exhibits almost no seasonality, while a series with strong seasonality will have $F_S$ close to 1 because $Var(R_t)$ will be much smaller than $Var(S_t + R_t)$.

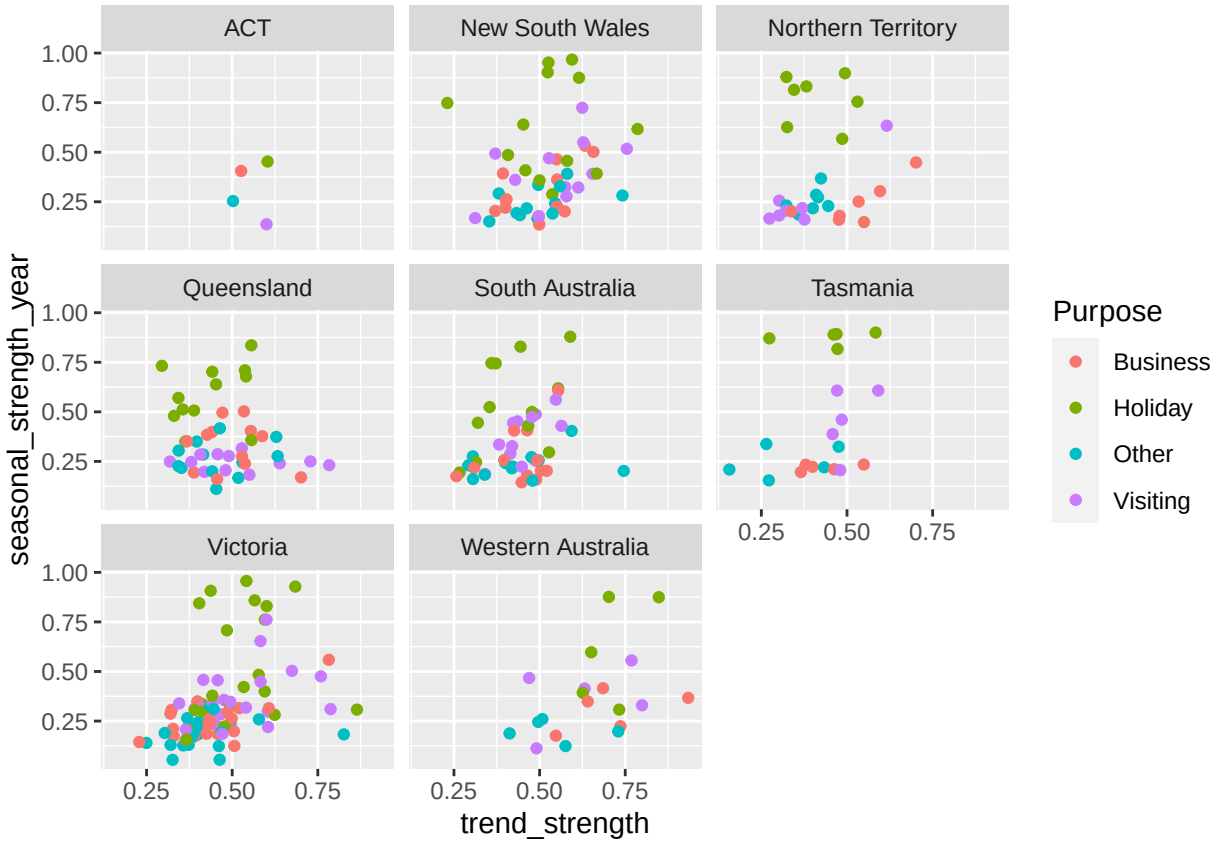
These measures can be useful, for example, when you have a large collection of time series, and you need to find the series with the most trend or the most seasonality. These and other STL-based features are computed using the feat_stl() function.

```
tourism |>
  features(Trips, feat_stl)
```

```
## # A tibble: 304 x 12
##    Region State Purpose trend_strength seasonal_strength_year seasonal_peak_year
##    <chr>  <chr> <chr>            <dbl>                  <dbl>              <dbl>
##  1 Adela~ Sout~ Busine~          0.464                  0.407                  3
##  2 Adela~ Sout~ Holiday          0.554                  0.619                  1
##  3 Adela~ Sout~ Other            0.746                  0.202                  2
##  4 Adela~ Sout~ Visiti~          0.435                  0.452                  1
##  5 Adela~ Sout~ Busine~          0.464                  0.179                  3
##  6 Adela~ Sout~ Holiday          0.528                  0.296                  2
##  7 Adela~ Sout~ Other            0.593                  0.404                  2
##  8 Adela~ Sout~ Visiti~          0.488                  0.254                  0
##  9 Alice~ Nort~ Busine~          0.534                  0.251                  0
## 10 Alice~ Nort~ Holiday          0.381                  0.832                  3
## # i 294 more rows
## # i 6 more variables: seasonal_trough_year <dbl>, spikiness <dbl>,
## #   linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>
```

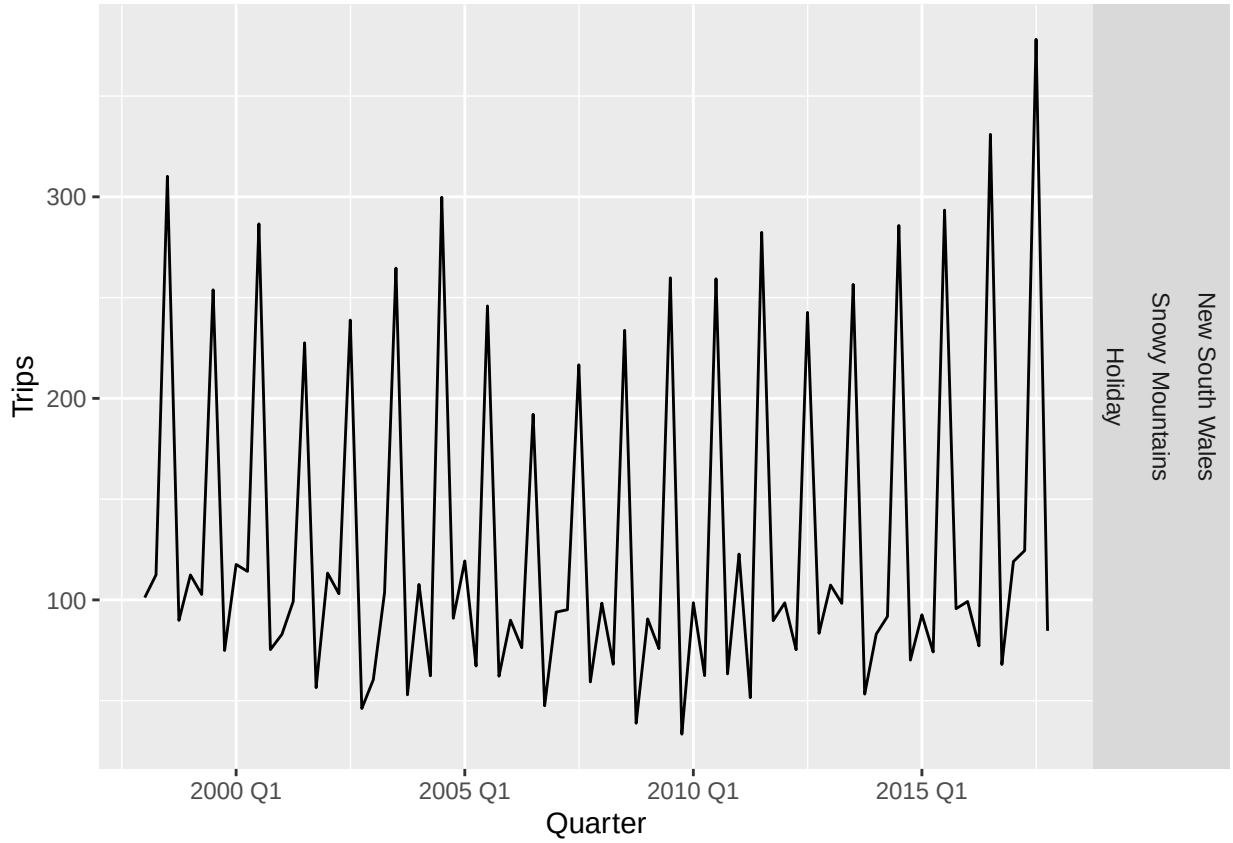
We can then use these features in plots to identify what type of series are heavily trended and what are most seasonal.

```
tourism |>
  features(Trips, feat_stl) |>
  ggplot(aes(x = trend_strength, y = seasonal_strength_year,
             col = Purpose)) +
  geom_point() +
  facet_wrap(vars(State))
```




Clearly, holiday series are most seasonal which is unsurprising. The strongest trends tend to be in Western Australia and Victoria. The most seasonal series can also be easily identified and plotted.

```
tourism |>
  features(Trips, feat_stl) |>
  filter(
    seasonal_strength_year == max(seasonal_strength_year)
  ) |>
  left_join(tourism, by = c("State", "Region", "Purpose"), multiple = "all") |>
  ggplot(aes(x = Quarter, y = Trips)) +
  geom_line() +
  facet_grid(vars(State, Region, Purpose))
```




This shows holiday trips to the most popular ski region of Australia.

The feat_stl() function returns several more features other than those discussed above.

- `seasonal_peak_year` indicates the timing of the peaks — which month or quarter contains the largest seasonal component. This tells us something about the nature of the seasonality. In the Australian tourism data, if Quarter 3 is the peak seasonal period, then people are travelling to the region in winter, whereas a peak in Quarter 1 suggests that the region is more popular in summer.
- `seasonal_trough_year` indicates the timing of the troughs — which month or quarter contains the smallest seasonal component.
- `spikiness` measures the prevalence of spikes in the remainder component $R_t$ of the STL decomposition. It is the variance of the leave-one-out variances of $R_t$.
- `linearity` measures the linearity of the trend component of the STL decomposition. It is based on the coefficient of a linear regression applied to the trend component.
- `curvature` measures the curvature of the trend component of the STL decomposition. It is based on the coefficient from an orthogonal quadratic regression applied to the trend component.
- `stl_e_acf1` is the first autocorrelation coefficient of the remainder series.
- `stl_e_acf10` is the sum of squares of the first ten autocorrelation coefficients of the remainder series.

## 4.4 Other features

Many more features are possible, and the `feasts` package computes only a few dozen features that have proven useful in time series analysis. It is also easy to add your own features by writing an R function that

takes a univariate time series input and returns a numerical vector containing the feature values.

The remaining features in the `feasts` package, not previously discussed, are listed here for reference. The details of some of them are discussed later in the book.

- `coef_hurst` will calculate the Hurst coefficient of a time series which is a measure of "long memory". A series with long memory will have significant autocorrelations for many lags.
- `feat_spectral` will compute the (Shannon) spectral entropy of a time series, which is a measure of how easy the series is to forecast. A series which has strong trend and seasonality (and so is easy to forecast) will have entropy close to 0. A series that is very noisy (and so is difficult to forecast) will have entropy close to 1.
- `box_pierce` gives the Box-Pierce statistic for testing if a time series is white noise, and the corresponding p-value. This test is discussed in Section 5.4.
- `ljung_box` gives the Ljung-Box statistic for testing if a time series is white noise, and the corresponding p-value. This test is discussed in Section 5.4.
- The $k^{th}$ partial autocorrelation measures the relationship between observations $k$ periods apart after removing the effects of observations between them. So the first partial autocorrelation ($k = 1$) is identical to the first autocorrelation, because there is nothing between consecutive observations to remove. Partial autocorrelations are discussed in Section 9.5. The `feat_pacf` function contains several features involving partial autocorrelations including the sum of squares of the first five partial autocorrelations for the original series, the first-differenced series and the second-differenced series. For seasonal data, it also includes the partial autocorrelation at the first seasonal lag.
- `unitroot_kpss` gives the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) statistic for testing if a series is stationary, and the corresponding p-value. This test is discussed in Section 9.1.
- `unitroot_pp` gives the Phillips-Perron statistic for testing if a series is non-stationary, and the corresponding p-value.
- `unitroot_ndiffs` gives the number of differences required to lead to a stationary series based on the KPSS test. This is discussed in Section 9.1
- `unitroot_nsdiffs` gives the number of seasonal differences required to make a series stationary. This is discussed in Section 9.1.
- `var_tiled_mean` gives the variances of the "tiled means" (i.e., the means of consecutive non-overlapping blocks of observations). The default tile length is either 10 (for non-seasonal data) or the length of the seasonal period. This is sometimes called the "stability" feature.
- `var_tiled_var` gives the variances of the "tiled variances" (i.e., the variances of consecutive non-overlapping blocks of observations). This is sometimes called the "lumpiness" feature.
- `shift_level_max` finds the largest mean shift between two consecutive sliding windows of the time series. This is useful for finding sudden jumps or drops in a time series.
- `shift_level_index` gives the index at which the largest mean shift occurs. shift_var_max finds the largest variance shift between two consecutive sliding windows of the time series. This is useful for finding sudden changes in the volatility of a time series.
- `shift_var_index` gives the index at which the largest variance shift occurs.
- `shift_kl_max` finds the largest distributional shift (based on the Kulback-Leibler divergence) between two consecutive sliding windows of the time series. This is useful for finding sudden changes in the distribution of a time series.
- `shift_kl_index` gives the index at which the largest KL shift occurs.

- `n_crossing_points` computes the number of times a time series crosses the median.
- `longest_flat_spot` computes the number of sections of the data where the series is relatively unchanging.
- `stat_arch_lm` returns the statistic based on the Lagrange Multiplier (LM) test of Engle (1982) for autoregressive conditional heteroscedasticity (ARCH).
- `guerrero` computes the optimal $\lambda$ value for a Box-Cox transformation using the Guerrero method (discussed in Section 3.1).

## 4.5 Exploring Australian tourism data

All of the features included in the feasts package can be computed in one line like this.

```
tourism_features <- tourism |>
  features(Trips, feature_set(pkgs = "feasts"))
```

```
## Warning: `n_flat_spots()` was deprecated in feasts 0.1.5.
## i Please use `longest_flat_spot()` instead.
## i The deprecated feature was likely used in the fabletools package.
##   Please report the issue at <https://github.com/tidyverts/fabletools/issues>.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
tourism_features
```

```
## # A tibble: 304 x 51
##    Region State Purpose trend_strength seasonal_strength_year seasonal_peak_year
##    <chr>  <chr> <chr>            <dbl>                  <dbl>              <dbl>
##  1 Adela~ Sout~ Busine~          0.464                  0.407                  3
##  2 Adela~ Sout~ Holiday          0.554                  0.619                  1
##  3 Adela~ Sout~ Other            0.746                  0.202                  2
##  4 Adela~ Sout~ Visiti~          0.435                  0.452                  1
##  5 Adela~ Sout~ Busine~          0.464                  0.179                  3
##  6 Adela~ Sout~ Holiday          0.528                  0.296                  2
##  7 Adela~ Sout~ Other            0.593                  0.404                  2
##  8 Adela~ Sout~ Visiti~          0.488                  0.254                  0
##  9 Alice~ Nort~ Busine~          0.534                  0.251                  0
## 10 Alice~ Nort~ Holiday          0.381                  0.832                  3
## # i 294 more rows
## # i 45 more variables: seasonal_trough_year <dbl>, spikiness <dbl>,
## #   linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>,
## #   acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>, pacf5 <dbl>,
## #   diff1_pacf5 <dbl>, diff2_pacf5 <dbl>, season_pacf <dbl>,
## #   zero_run_mean <dbl>, nonzero_squared_cv <dbl>, zero_start_prop <dbl>, ...
```

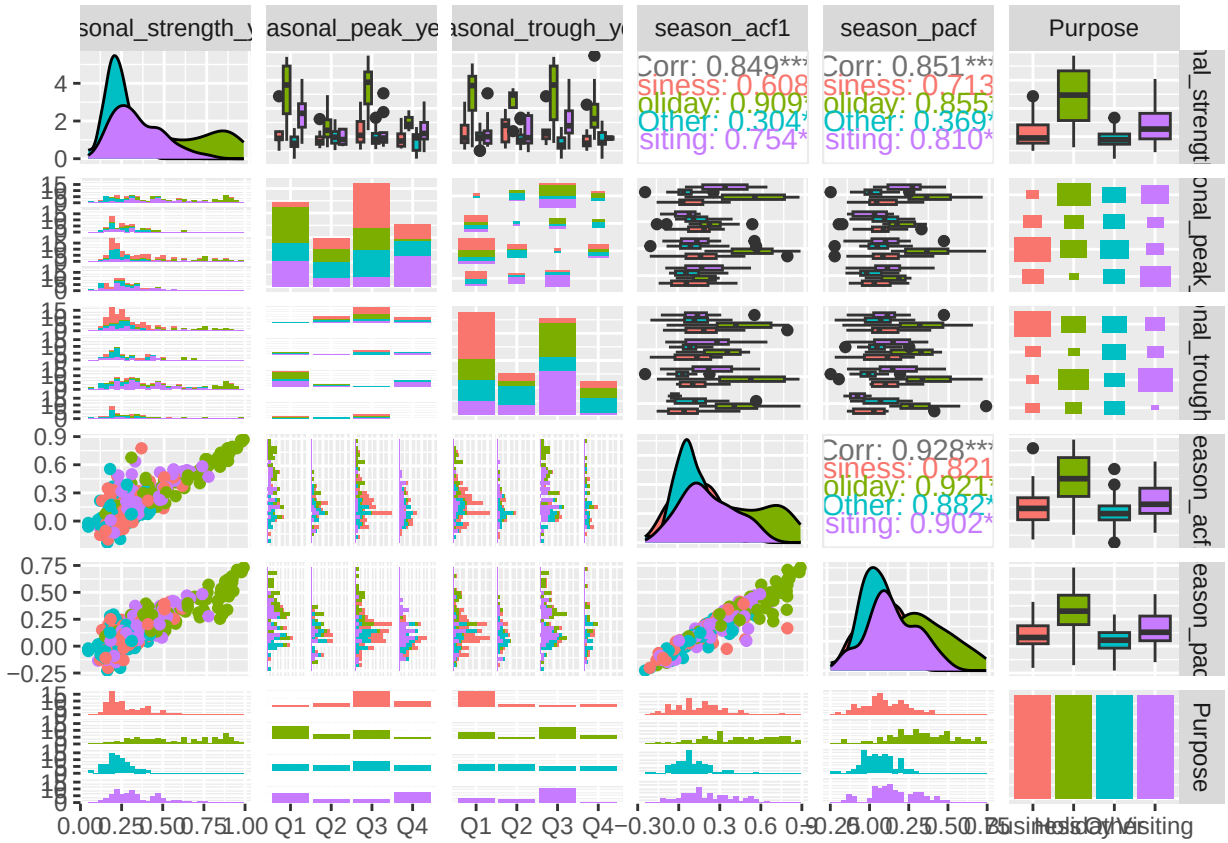
Provided the `urca` and `fracdiff` packages are installed, this gives 48 features for every combination of the three key variables (`Region`, `State` and `Purpose`). We can treat this tibble like any data set and analyse it to find interesting observations or groups of observations.

We've already seen how we can plot one feature against another (Section 4.3). We can also do pairwise plots of groups of features. In Figure 4.3, for example, we show all features that involve seasonality, along with the `Purpose` variable.

```
library(glue)
tourism_features |>
  select_at(vars(contains("season"), Purpose)) |>
  mutate(
    seasonal_peak_year = seasonal_peak_year +
      4*(seasonal_peak_year==0),
    seasonal_trough_year = seasonal_trough_year +
      4*(seasonal_trough_year==0),
    seasonal_peak_year = glue("Q{seasonal_peak_year}"),
    seasonal_trough_year = glue("Q{seasonal_trough_year}"),
  ) |>
  GGally::ggpairs(mapping = aes(colour = Purpose))
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

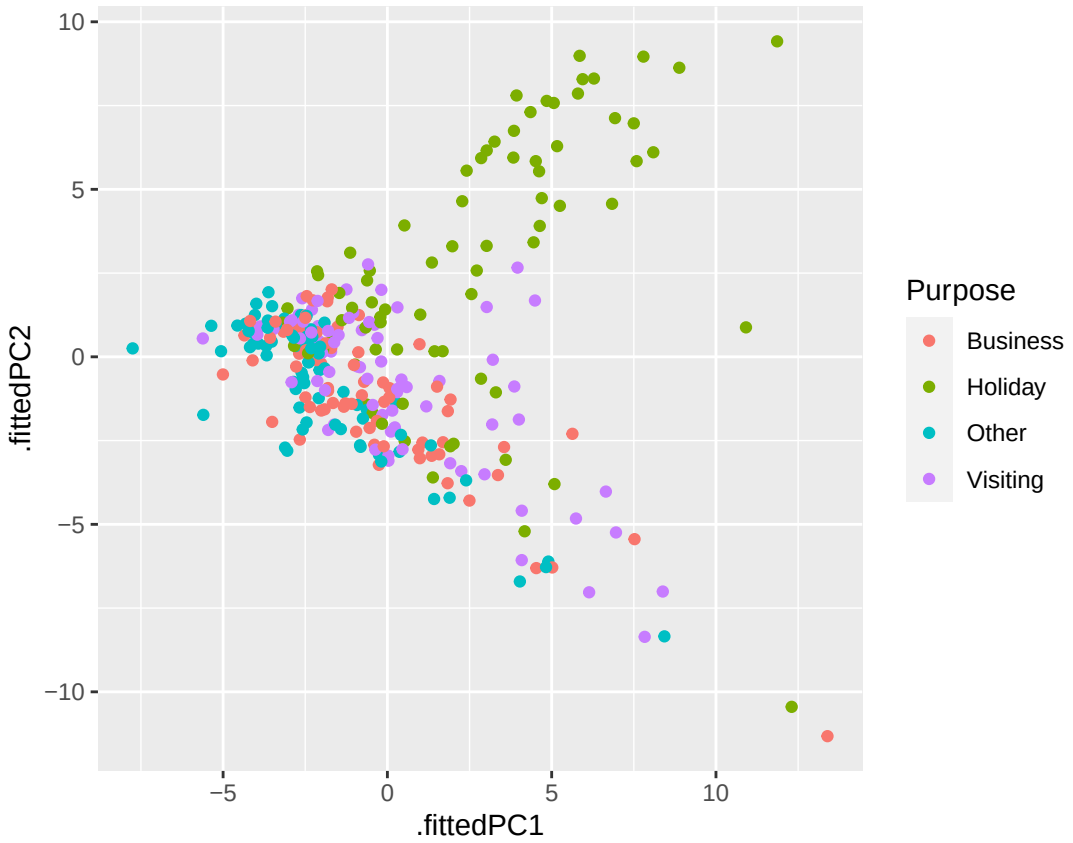



Here, the Purpose variable is mapped to colour. There is a lot of information in this figure, and we will highlight just a few things we can learn.

- The three numerical measures related to seasonality (seasonal_strength_year, season_acf1 and season_pacf) are all positively correlated.
- The bottom left panel and the top right panel both show that the most strongly seasonal series are related to holidays (as we saw previously).
- The bar plots in the bottom row of the seasonal_peak_year and seasonal_trough_year columns show that seasonal peaks in Business travel occur most often in Quarter 3, and least often in Quarter 1.

It is difficult to explore more than a handful of variables in this way. A useful way to handle many more variables is to use a dimension reduction technique such as principal components. This gives linear combinations of variables that explain the most variation in the original data. We can compute the principal components of the tourism features as follows.

```
library(broom)
pcs <- tourism_features |>
  select(-State, -Region, -Purpose) |>
  prcomp(scale = TRUE) |>
  augment(tourism_features)
pcs |>
  ggplot(aes(x = .fittedPC1, y = .fittedPC2, col = Purpose)) +
  geom_point() +
  theme(aspect.ratio = 1)
```




Each point on Figure 4.4 represents one series and its location on the plot is based on all 48 features. The first principal component (.fittedPC1) is the linear combination of the features which explains the most variation in the data. The second principal component (.fittedPC2) is the linear combination which explains the next most variation in the data, while being uncorrelated with the first principal component. For more information about principal component dimension reduction, see Izenman (2008).

Figure 4.4 reveals a few things about the tourism data. First, the holiday series behave quite differently from the rest of the series. Almost all of the holiday series appear in the top half of the plot, while almost all of the remaining series appear in the bottom half of the plot. Clearly, the second principal component is distinguishing between holidays and other types of travel.

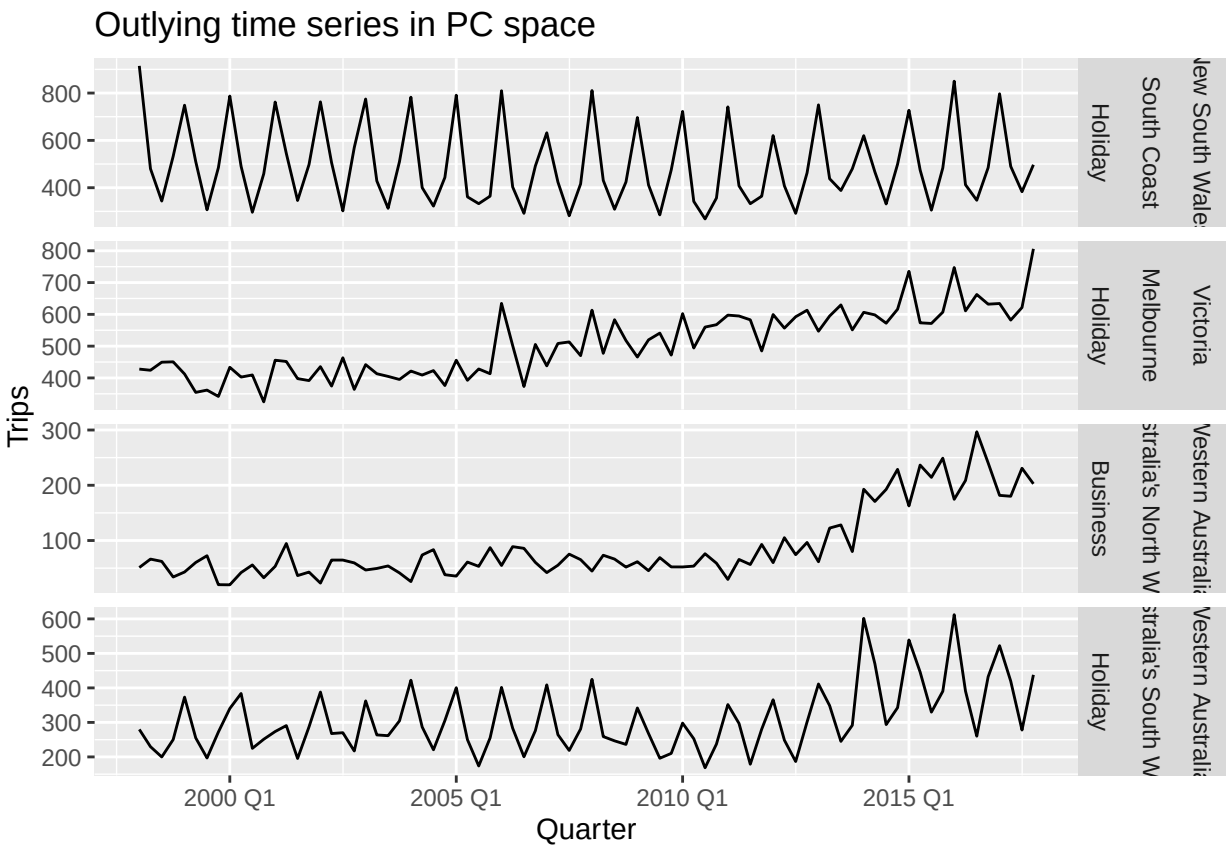
The plot also allows us to identify anomalous time series — series which have unusual feature combinations. These appear as points that are separate from the majority of series in Figure 4.4. There are four that stand out, and we can identify which series they correspond to as follows.

```
outliers <- pcs |>
  filter(.fittedPC1 > 10) |>
  select(Region, State, Purpose, .fittedPC1, .fittedPC2)
outliers
```

```
## # A tibble: 4 x 5
##   Region                 State            Purpose  .fittedPC1 .fittedPC2
##   <chr>                  <chr>            <chr>         <dbl>      <dbl>
## 1 Australia's North West Western Australia Business       13.4    -11.3
## 2 Australia's South West Western Australia Holiday        10.9      0.880
## 3 Melbourne              Victoria          Holiday        12.3    -10.4
## 4 South Coast            New South Wales   Holiday        11.9      9.42
```

```
#> # A tibble: 4 × 5
#>   Region                  State             Purpose  .fittedPC1 .fittedPC2
#>   <chr>                   <chr>             <chr>         <dbl>      <dbl>
#> 1 Australia's North West Western Australia Business       13.4     -11.3
#> 2 Australia's South West Western Australia Holiday        10.9       0.880
#> 3 Melbourne              Victoria          Holiday        12.3     -10.4
#> 4 South Coast            New South Wales   Holiday        11.9       9.42
```

```
outliers |>
  left_join(tourism, by = c("State", "Region", "Purpose"), multiple = "all") |>
  mutate(Series = glue("{State}", "{Region}", "{Purpose}", .sep = "\n\n")) |>
  ggplot(aes(x = Quarter, y = Trips)) +
  geom_line() +
  facet_grid(Series ~ ., scales = "free") +
  labs(title = "Outlying time series in PC space")
```




We can speculate why these series are identified as unusual.

- Holiday visits to the south coast of NSW is highly seasonal but has almost no trend, whereas most holiday destinations in Australia show some trend over time.
- Melbourne is an unusual holiday destination because it has almost no seasonality, whereas most holiday destinations in Australia have highly seasonal tourism.
- The north western corner of Western Australia is unusual because it shows an increase in business tourism in the last few years of data, but little or no seasonality.

- The south western corner of Western Australia is unusual because it shows both an increase in holiday tourism in the last few years of data and a high level of seasonality.