In the [last post](https://robjhyndman.com/hyndsight/feasts/), We showed how the feasts package can be used to produce various time series graphics.

The feasts package also includes functions for computing FEatures And Statistics from Time Series (hence the name). In this post I will give three examples of how these might be used.

library(tidyverse)

library(tsibble)

library(feasts)

**Exploring Australian tourism data**

I used this example in [my talk at useR!2019 in Toulouse](https://robjhyndman.com/seminars/isf-feasts/), and it is also the basis of [a vignette in the package](http://feasts.tidyverts.org/articles/feasts.html), and a recent [blog post by Mitchell O’Hara-Wild](https://blog.mitchelloharawild.com/blog/feasts/). The data set contains domestic tourist visitor nights in Australia, disaggregated by State, Region and Purpose.

tourism

## # A tsibble: 24,320 x 5 [1Q]

## # Key: Region, State, Purpose [304]

## Quarter Region State Purpose Trips

##

## 1 1998 Q1 Adelaide South Australia Business 135.

## 2 1998 Q2 Adelaide South Australia Business 110.

## 3 1998 Q3 Adelaide South Australia Business 166.

## 4 1998 Q4 Adelaide South Australia Business 127.

## 5 1999 Q1 Adelaide South Australia Business 137.

## 6 1999 Q2 Adelaide South Australia Business 200.

## 7 1999 Q3 Adelaide South Australia Business 169.

## 8 1999 Q4 Adelaide South Australia Business 134.

## 9 2000 Q1 Adelaide South Australia Business 154.

## 10 2000 Q2 Adelaide South Australia Business 169.

## # … with 24,310 more rows

An example of a feature would be the autocorrelation function at lag 1 — it is a numerical summary capturing some aspect of the time series. Autocorrelations at other lags are also features, as are the autocorrelations of the first differenced series, or the seasonally differenced series, etc. Another example of a feature is the strength of seasonality of a time series, as measured by \(1-\text{Var}(R\_t)/\text{Var}(S\_t+R\_t)\) where \(S\_t\) is the seasonal component and \(R\_t\) is the remainder component in an STL decomposition. Values close to 1 indicate a highly seasonal time series, while values close to 0 indicate a time series with little seasonality.

The feasts package has some inbuilt feature calculation functions. For example coef\_hurst will calculate the Hurst coefficient of a time series and feat\_spectral will compute the spectral entropy of a time series. To apply these to all series in the tourism data set, we can use the features function like this.

tourism %>% features(Trips, list(coef\_hurst, feat\_spectral))

## # A tibble: 304 x 5

## Region State Purpose coef\_hurst spectral\_entropy

##

## 1 Adelaide South Australia Business 0.571 0.988

## 2 Adelaide South Australia Holiday 0.558 0.968

## 3 Adelaide South Australia Other 0.862 0.928

## 4 Adelaide South Australia Visiting 0.583 0.964

## 5 Adelaide Hills South Australia Business 0.555 0.980

## 6 Adelaide Hills South Australia Holiday 0.651 0.986

## 7 Adelaide Hills South Australia Other 0.672 0.973

## 8 Adelaide Hills South Australia Visiting 0.653 0.994

## 9 Alice Springs Northern Territory Business 0.713 0.973

## 10 Alice Springs Northern Territory Holiday 0.500 0.826

## # … with 294 more rows

There are also functions which produce collections of features based on tags. For example, feature\_set(tags="acf") will produce 7 features based on various autocorrelation coefficients, while feature\_set(tags="stl") produces 7 features based on an STL decomposition of a time series, including the strength of seasonality mentioned above.

tourism %>% features(Trips, feature\_set(tags="stl"))

## # A tibble: 304 x 12

## Region State Purpose trend\_strength seasonal\_streng… seasonal\_peak\_y…

##

## 1 Adela… Sout… Busine… 0.451 0.380 3

## 2 Adela… Sout… Holiday 0.541 0.601 1

## 3 Adela… Sout… Other 0.743 0.189 2

## 4 Adela… Sout… Visiti… 0.433 0.446 1

## 5 Adela… Sout… Busine… 0.453 0.140 3

## 6 Adela… Sout… Holiday 0.512 0.244 2

## 7 Adela… Sout… Other 0.584 0.374 2

## 8 Adela… Sout… Visiti… 0.481 0.228 0

## 9 Alice… Nort… Busine… 0.526 0.224 0

## 10 Alice… Nort… Holiday 0.377 0.827 3

## # … with 294 more rows, and 6 more variables: seasonal\_trough\_year ,

## # spikiness , linearity , curvature , stl\_e\_acf1 ,

## # stl\_e\_acf10

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, feature\_set(tags="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.

The most seasonal series can also be easily identified and plotted.

tourism %>%

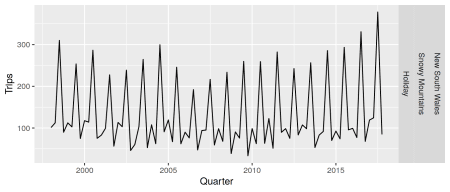
features(Trips, feature\_set(tags="stl")) %>%

filter(seasonal\_strength\_year == max(seasonal\_strength\_year)) %>%

left\_join(tourism, by = c("State","Region","Purpose")) %>%

ggplot(aes(x = Quarter, y = Trips)) + geom\_line() +

facet\_grid(vars(State,Region,Purpose))



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

The feasts package currently includes 42 features which can all be computed in one line like this.

# Compute features

tourism\_features <- tourism %>%

features(Trips, feature\_set(pkgs="feasts"))

Unusual time series can be identified by first doing a principal components decomposition

pcs <- tourism\_features %>%

select(-State, -Region, -Purpose) %>%

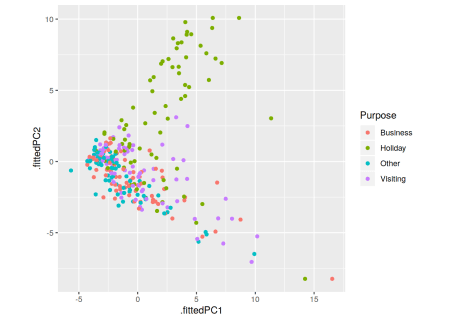
prcomp(scale=TRUE) %>%

augment(tourism\_features)

pcs %>%

ggplot(aes(x=.fittedPC1, y=.fittedPC2, col=Purpose)) +

geom\_point() + theme(aspect.ratio=1)



We can then identify some unusual series.

pcs %>%

filter(.fittedPC1 > 10.5) %>%

select(Region, State, Purpose, .fittedPC1, .fittedPC2)

## # A tibble: 3 x 5

## Region State Purpose .fittedPC1 .fittedPC2

##

## 1 Australia's North West Western Australia Business 16.5 -8.24

## 2 Australia's South West Western Australia Holiday 11.3 3.03

## 3 Melbourne Victoria Holiday 14.2 -8.24

**Replicating Hyndman, Wang and Laptev (2015)**

I used a similar approach to identifying outliers in my [ICDM 2015 paper with Earo Wang and Nikolay Laptev](https://robjhyndman.com/publications/icdm2015/). In that paper, we used hourly data on thousands of mail servers from Yahoo.

The data is stored in the tsfeatures package as an mts object. Unfortunately, mts and ts objects are particularly bad at handling hourly and other sub-daily data, as time is stored numerically and the precision is not sufficient to always work properly. So we need to do a little work to create a properly formulated tsibble.

yahoo <- tsfeatures::yahoo\_data() %>%

as\_tsibble() %>%

mutate(

Time = hms::hms(day = trunc(index) - 1L,

hour = as.integer((round(24\*(index-1))) %% 24))

) %>%

as\_tsibble(index = Time, key=key) %>%

select(Time, key, value)

The key variable here identifies the particular server and measurement variable on that server. There are 1748 such time series, each with 1437 (almost 60 days) observations.

Now we create the features on all series, matching the original paper as closely as possible. There are a few small differences due to different choices in how features are computed, but they make little difference to the results.

The features are computed in two groups — the first group uses the original data, while the second group uses the scaled data.

yahoo\_features <- bind\_cols(

yahoo %>% features(value, features=list(

mean = ~ mean(., na.rm = TRUE),

var = ~ var(., na.rm = TRUE)

)),

yahoo %>% features(scale(value), features = list(

~ feat\_acf(.),

~ feat\_spectral(.),

~ n\_flat\_spots(.),

~ n\_crossing\_points(.),

~ var\_tiled\_var(., .period = 24, .size = 24),

~ shift\_level\_max(., .period = 24, .size = 24),

~ shift\_var\_max(., .period = 24, .size = 24),

~ shift\_kl\_max(., .period = 24, .size = 48),

~ feat\_stl(., .period = 24, s.window = "periodic", robust = TRUE)

))

) %>%

rename(lumpiness = var\_tiled\_var) %>%

select(key, mean, var, acf1, trend\_strength,

seasonal\_strength\_24, linearity, curvature,

seasonal\_peak\_24, seasonal\_trough\_24,

spectral\_entropy, lumpiness, spikiness,

shift\_level\_max, shift\_var\_max,

n\_flat\_spots, n\_crossing\_points,

shift\_kl\_max, shift\_kl\_index)

Now we can compute the principal components.

hwl\_pca <- yahoo\_features %>%

select(-key) %>%

na.omit() %>%

prcomp(scale=TRUE) %>%

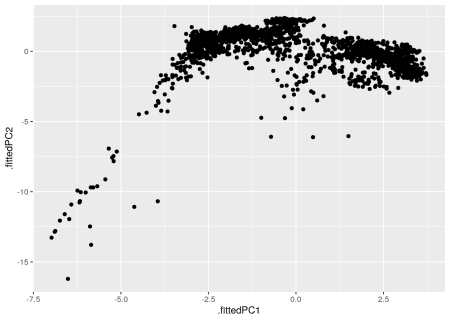
augment(na.omit(yahoo\_features))

hwl\_pca %>%

as\_tibble() %>%

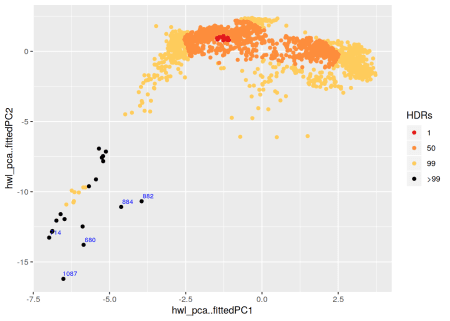
ggplot(aes(x=.fittedPC1, y=.fittedPC2)) +

geom\_point()



A useful plot for identifying the outlying observations is an HDR scatterplot which highlights regions of high density, and identifies outliers as those observations in low density regions. The red dots are the 1% highest density observations, the orange dots are in the 50% region, the yellow dots are in the 99% regions, while those shown in black are the 1% most “unusual” observations (having the lowest bivariate density on this plot.) The five most outlying points are highlighted by their row numbers.

hdrcde::hdrscatterplot(hwl\_pca$.fittedPC1, hwl\_pca$.fittedPC2, noutliers=5)



**Replicating Kang, Hyndman and Smith-Miles (2017)**

Another paper that uses features is this [IJF paper from 2017](https://robjhyndman.com/publications/ts-feature-space/) in which we explored the feature space of the M3 time series data.

First we need to create several tsibbles, corresponding to the different seasonal periods in the M3 data.

m3totsibble <- function(z) {

rbind(

as\_tsibble(z$x) %>% mutate(Type="Training"),

as\_tsibble(z$xx) %>% mutate(Type="Test")

) %>%

mutate(

st = z$st,

type = z$type,

period = z$period,

description = z$description,

sn = z$sn,

) %>%

as\_tibble()

}

m3yearly <- Mcomp::M3 %>%

subset("yearly") %>%

purrr::map\_dfr(m3totsibble) %>%

as\_tsibble(index=index, key=c(sn,period,st))

m3quarterly <- Mcomp::M3 %>%

subset("quarterly") %>%

purrr::map\_dfr(m3totsibble) %>%

mutate(index = yearquarter(index)) %>%

as\_tsibble(index=index, key=c(sn,period,st))

m3monthly <- Mcomp::M3 %>%

subset("monthly") %>%

purrr::map\_dfr(m3totsibble) %>%

mutate(index = yearmonth(index)) %>%

as\_tsibble(index=index, key=c(sn,period,st))

m3other <- Mcomp::M3 %>%

subset("other") %>%

purrr::map\_dfr(m3totsibble) %>%

as\_tsibble(index=index, key=c(sn,period,st))

There are some bespoke features used by Kang et al (IJF 2017), so rather than use the inbuilt feasts functions, we can create our own feature calculation function.

khs\_stl <- function(x, period) {

output <- c(frequency=period, feat\_spectral(x))

lambda <- forecast::BoxCox.lambda(ts(x, frequency=period),

lower=0, upper=1, method='loglik')

stlfeatures <- feat\_stl(box\_cox(x, lambda), .period=period,

s.window='periodic', robust=TRUE)

output <- c(output, stlfeatures, lambda=lambda)

if(period==1L)

output <- c(output, seasonal\_strength=0)

return(output)

}

m3\_features <- bind\_rows(

m3yearly %>% features(value, features=list(~ khs\_stl(.,1))),

m3quarterly %>% features(value, features=list(~ khs\_stl(.,4))) %>%

rename(seasonal\_strength = seasonal\_strength\_4),

m3monthly %>% features(value, features=list(~ khs\_stl(.,12))) %>%

rename(seasonal\_strength = seasonal\_strength\_12),

m3other %>% features(value, features=list(~ khs\_stl(.,1)))

) %>%

select(sn, spectral\_entropy, trend\_strength, seasonal\_strength,

frequency, stl\_e\_acf1, lambda)

Finally we plot the first two principal components of the feature space.

m3\_features %>%

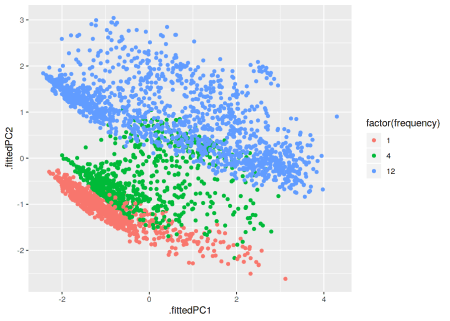
select(-sn) %>%

prcomp(scale=TRUE) %>%

augment(m3\_features) %>%

ggplot(aes(x=.fittedPC1, y=.fittedPC2)) +

geom\_point(aes(color = factor(frequency)))



To **leave a comment** for the author, please follow the link and comment on their blog: [**R on Rob J Hyndman**](https://robjhyndman.com/hyndsight/fbtsa/).