PATH Forecast Documentation

Planning & Regional Development

6/2/2020

DRAFT

Summary

Discussion across staff regarding PATH activity models and forecasts generally use the term "model." There are actually three models, one apiece for weekday, Saturday and Sunday ridership.[1] This file attempts to document those models and the process of using them to produce the forecast in a manner that allows for transparency and reproducibility across the agency and, as needed, with outside parties.

The three models are "trained" using the data available through early June. The forward-looking forecasting process then proceeds by subjecting future scenarios for the covariates to those equations and producing predicted values.

This file includes most relevant source code included in the forecast development process, and the full code is provided as an attached source file in the transmittal folder. This is done for transparency, and we have tried to minimize font associated with code to make it easy to distinguish between it and the narrative describing the process. All source code used, be it to import data or to merge files and export results, is included in the attached source file.

The economics data, already converted to monthly, is combined with days, dummy variables (simulating past events that need special statistical controls), ridership, and fares. They're saved across a handful of small worksheets and are merged in-memory. The data is thus almost completely unaltered prior to actual analysis (model estimation and forecasting). The exception is the fare variable, which is converted from nominal to real using the most recent macroeconomic forecast's value for national CPI.

PATH forecasts through early 2020 used regional CPI, and going forward will use national CPI to conform with TB&T forecasting and agency financial analysis.

PATH and Planning developed the general outlines for the models three years ago. Specifications since then have not changed, outside of the allowance of shifting time series controls.[2] The weekday model, which easily forecasts the largest share of total ridership, has performed generally well but the weekend models have gradually lost predictive power as weekend closures have provided near-useless data points in increasing frequency. PATH and Planning attempted to address this challenge in 2019 and it remains unresolved.

The only other major change of note since the 2017 model development process is a shift in the definition of day types. Holidays were at one point all classified and treated as Sundays; the shift to the current treatment had a minor impact on the forecast.

The file has four sections. First, the summary (above). Second, code documentation. This is included for transparency and reproducibility, and can be moved to a later section in subsequent versions; for now, it is easiest to keep it as the second section. It can be used, for example, to find what file names are imported to and exported from the process. Third, output. This focuses on annual forecasted ridership. Fourth, modeling statistics and diagnostics. Modelers can review this information at their interest.

[1] Holidays are distributed across weekdays (for minor holidays, where PATH has found similarities between weekday and minor holiday ridership behavior) and Saturdays (major holidays). Weekdays include one minor holiday apiece in October (Columbus Day) and November (Veterans Day). Saturdays include other holidays. Sundays represent only Sunday ridership.

[2] The models' time series controls are reset roughly once a year, with guidance provided by automated variable selection algorithms. Nerds, see: https://otexts.com/fpp2/arima-r.html

Code

```
# Save economic monthly variables as:
write.csv(econ_month, "./input data/econ_vars_months 2020_06.csv")
```

```
days = read.csv("./input data/dates_dummies.csv")
days$month = as.Date(days$month, "%m/%d/%y")

path = read.csv("./input data/PATH input 2020_06.csv")
path$month = as.Date(days$month, "%m/%d/%y")

fare = read.csv("./input data/fare_nominal.csv")
fare$month = as.Date(fare$month, "%m/%d/%y")
```

```
# Merge
path = merge(days, path)  #,by='month')
path = merge(path, fare)  #,by='month')
path = merge(path, econ_month)
```

Details for real fare:[3]

```
path$cpi_base = path[path$month == end, "cpi_2020_06"]
path$real_farefare = ifelse(path$month <= end, path$fare_nominal *
    path$cpi_base/path$cpi_2020_06, max(path$fare_nominal))
path$cpi_base = NULL
```

```
before = subset(path, path$month <= end & path$month >= "2004-01-01")  #before = head(path, 218)
after = subset(path, path$month > end)  #after = tail(path, 250)
```

The code for training the models are below, starting with a quick list of the variables chosen for the three equations, one apiece for weekday, Saturday and Sunday ridership. Note that the Saturday and Sunday equations use the same varibles. Key predictors are Manhattan and Hudson County employment for the weekday model and Hudson County population for the weekend models.

Note: Models are trained below

Weekday:

```
fit = arima(ts(before$avg_wkdayholminor_tstile), xreg = oldreg,
    order = c(0, 0, 1), include.mean = T)
```

Saturday:

```
fitsat = arima(ts(before$avg_satholmajor_tstile), xreg = oldregsat,
    order = c(1, 1, 0))
```

Sunday:

[3] https://stackoverflow.com/questions/25646333/code-chunk-font-size-in-rmarkdown-with-knitr-and-latex/57151528#57151528

```
fitsun = arima(ts(before$avg_sun_tstile), xreg = oldregsat, order = c(1,
    1, 1))
```

```
pathpredict = predict(fit, n.ahead = forec_horizon, newxreg = newreg)  # level=95 #interval = 'predicti
pathpredictsat = predict(fitsat, n.ahead = forec_horizon, newxreg = newregsat)
pathpredictsun = predict(fitsun, n.ahead = forec_horizon, newxreg = newregsat)
```

```
pathpredict_by_month = as.data.frame(cbind(pathpredict$pred,
    pathpredictsat$pred, pathpredictsun$pred))
names(pathpredict_by_month) = c("avg_wkdayholminor_tstile", "avg_satholmajor_tstile",
    "avg_sun_tstile")

pathpredict_by_month$month = seq(as.Date(end) + extra, as.Date(future),
    by = "mon")

## Add old data (January 2017, for example) back to the pile.
before_mini = data.frame(before$month, (before$avg_wkdayholminor_tstile *
    before$num_wkdayholminor), (before$avg_satholmajor_tstile *
    before$num_satholmajor), (before$avg_sun_tstile * before$num_sun))
names(before_mini) = c("month", "avg_wkdayholminor_tstile", "avg_satholmajor_tstile",
    "avg_sun_tstile")

## Now multiply by number of days per month ...
pathpredict_by_month = merge(pathpredict_by_month, days)
pathpredict_by_month$sum_wkdayholminor = pathpredict_by_month$avg_wkdayholminor_tstile *
    pathpredict_by_month$num_wkdayholminor
pathpredict_by_month$sum_satholmajor = pathpredict_by_month$avg_satholmajor_tstile *
    pathpredict_by_month$num_satholmajor
pathpredict_by_month$sum_sun = pathpredict_by_month$avg_sun_tstile *
    pathpredict_by_month$num_sun
```

```
# Annual
pathpredict_by_month$year = year(pathpredict_by_month$month)
pathpredict_year = summaryBy(sum_wkdayholminor + sum_satholmajor +
    sum_sun ~ year, data = pathpredict_by_month, FUN = sum)
names(pathpredict_year) = c("year", "sum_wkdayholminor", "sum_satholmajor",
    "sum_sun")
pathpredict_year$total = pathpredict_year$sum_wkdayholminor +
    pathpredict_year$sum_satholmajor + pathpredict_year$sum_sun
pathpredict_by_month$year = NULL
```

```
resids = as.data.frame(cbind(as.vector(resid(fit)), as.vector(resid(fitsat)),
    as.vector(resid(fitsun))))
names(resids) = c("Weekday_residuals", "Saturday_residuals",
    "Sunday_residuals")
```

Output[4] [5]

Table 1: Annual Results

year	sum_wkdayholminor	sum_satholmajor	sum_sun	total
2020	62061215	5940558	4140258	72142030
2021	73384943	7485803	4858777	85729523
2022	73774253	7551378	4950881	86276511
2023	73551137	7709341	5136734	86397212
2024	74267564	7870779	5148917	87287260
2025	74250514	8021424	5229041	87500979
2026	74640412	8161811	5310211	88112433
2027	74806555	8454870	5391128	88652553
2028	75631252	8488777	5580311	89700339
2029	76266322	8636793	5575057	90478171
2030	76986557	8768208	5649963	91404728
2031	77763694	8851806	5692607	92308107
2032	78535868	9061348	5729872	93327088
2033	79284400	9003041	5762256	94049697
2034	79710261	9048934	5899456	94658651
2035	80717331	9089704	5827651	95634686
2036	81639452	9152864	5860407	96652722
2037	81678161	9363719	5981027	97022907
2038	81629348	9879068	6269191	97777608
2039	81034661	10820193	6795340	98650195
2040	79787805	12277569	7611180	99676554

Save everything as:

```
sheets = list(Data = path, Monthly_Output = pathpredict_by_month,
    Annual_Output = pathpredict_year, Residuals = resids)
write_xlsx(sheets, "./Output 2020-06.xlsx")
```

[4] Good source for tables in Markdown: https://rmarkdown.rstudio.com/lesson-7.html

[5] Exporting to Excel: openxlsx or writexl https://stackoverflow.com/questions/27713310/easy-way-to-export-multiple-data-frame-to-multiple-excel-worksheets

Modeling statistics and diagnostics

Weekday

Table 2: Weekday Coefficients

term	estimate	std.error
ma1	0.5818412	0.046948
intercept	-75024.9290748	23948.678344
before.man_hud	134.5121663	14.056278
before.dummy_2	5130.0952567	3352.115155
before.dummy_3	7693.8249226	4469.531990
before.dummy_4	13184.6786762	4493.298293
before.dummy_5	16036.3324675	4490.148783
before.dummy_6	19450.7738027	4488.246987
before.dummy_7	15497.4014653	4487.284963
before.dummy_8	9473.2767975	4486.973176
before.dummy_9	19941.9137549	4487.364169
before.dummy_10	17501.6970053	4488.724629
before.dummy_11	10821.5045909	4534.850687
before.dummy_12	335.8490213	3432.823634
before.dum_911_base	-23445.0830732	11431.889695
before.supersandy	-74600.1634651	10185.084418
before.real_farefare	-13404.0442636	6487.789700

Table 3: Weekday Diagnostics

sigma	logLik	AIC	BIC
11008.21	-2080.922	4197.843	4256.665

Table 4: Weekday Additional Diagnostics

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	3.452376	11008.21	8303.782	-0.2936778	3.617164	1.093965	0.3012808

Saturday

Table 5: Saturday Coefficients

term	estimate	std.error
ar1	-0.5650652	0.0596943
before.pop_hudson	525.8008788	653.1165540
before.dummy_2	6791.0034876	1801.2989246
before.dummy_3	20294.9602632	1653.5031576
before.dummy_4	22283.5192317	1996.8108742
before.dummy_5	14122.0928316	1984.2442145
before.dummy_6	21119.3960890	2107.0998529
before.dummy_7	17176.2970951	2077.6371181
before.dummy_8	15862.6316632	2119.4924092
before.dummy_9	16966.5411090	1995.5441919
before.dummy_10	20745.0956404	2019.0795177
before.dummy_11	14761.0625689	1664.5181296
before.dummy_12	15645.2205534	1865.6373499
before.dum_911_base	923.0360039	6270.7169378
before.supersandy	-44141.6380424	4150.5403567
before.end_close	-12894.5854535	2115.7885230
before.real_farefare	-24838.0113683	9679.7968393

Table 6: Saturday Diagnostics

sigma	logLik	AIC	BIC
6066.214	-1955.172	3946.344	4005.072

Table 7: Saturday Additional Diagnostics

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	64.49915	6050.575	4727.134	-0.1680862	4.454908	0.5536712	-0.1688497

Sunday

Table 8: Sunday Coefficients

term	estimate	std.error
ar1	-0.0429121	0.0874589
ma1	-0.7813257	0.0503897
before.pop_hudson	344.3567219	188.1226266
before.dummy_2	5192.8486378	1559.4431145
before.dummy_3	7995.6716411	1570.5232983
before.dummy_4	14010.0203149	1582.8441294
before.dummy_5	18515.3974745	1588.5338312
before.dummy_6	23926.2298915	1592.5920265
before.dummy_7	17475.9688344	1594.3872602
before.dummy_8	15930.8572114	1597.4277955
before.dummy_9	20927.1833836	1593.5113991
before.dummy_10	17499.1731507	1587.2040691
before.dummy_11	13877.2990934	1586.3684762
before.dummy_12	16593.4569808	1607.7736887
before.dum_911_base	-2997.4674056	5024.8089255
before.supersandy	-37281.9095052	3675.0330581
before.end_close	-10381.0619716	1409.0447399
before.real_farefare	-12262.3556354	5557.3500671

Table 9: Sunday Diagnostics

sigma	logLik	AIC	BIC
4876.593	-1913.355	3864.711	3926.702

Table 10: Sunday Additional Diagnostics

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	208.073	4864.018	3557.144	-0.0589807	4.54416	0.5178687	-0.0101914

Model residuals