Demonstration of Experiments (paper IJGI)

Ming Li

Tuesday, August 19, 2014

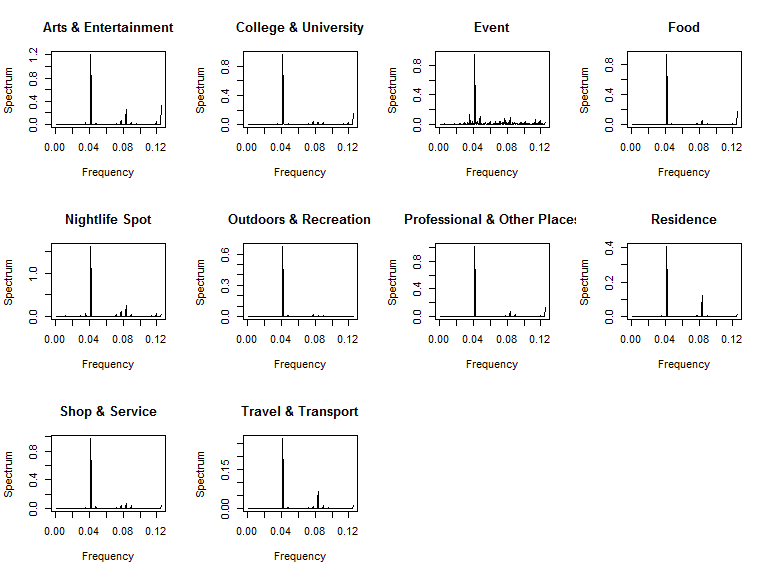
First, we load the libraries, functions, and data into the workspace.Following is a summary of the dataset.

## gid user\_id gender   
## Min. : 1 Min. : 32 female:235071   
## 1st Qu.: 254982 1st Qu.: 1514021 male :285587   
## Median : 506430 Median : 6820154 NA's : 7707   
## Mean : 505864 Mean :15433981   
## 3rd Qu.: 756951 3rd Qu.:21518413   
## Max. :1135513 Max. :88116325   
##   
## venue\_id lat.x lon.x   
## 43a52546f964a520532c1fe3: 11454 Min. :40.5 Min. :-74.2   
## 4ace6c89f964a52078d020e3: 7211 1st Qu.:40.7 1st Qu.:-74.0   
## 42911d00f964a520f5231fe3: 3224 Median :40.7 Median :-74.0   
## 4ae6363ef964a520aba521e3: 2886 Mean :40.7 Mean :-74.0   
## 49b7ed6df964a52030531fe3: 2518 3rd Qu.:40.8 3rd Qu.:-74.0   
## 42829c80f964a5206a221fe3: 1999 Max. :40.9 Max. :-73.7   
## (Other) :499073   
## timestamps.x cate\_l1   
## Min. :1.39e+09 Food :155557   
## 1st Qu.:1.39e+09 Shop & Service : 83006   
## Median :1.40e+09 Nightlife Spot : 63810   
## Mean :1.40e+09 Travel & Transport : 56886   
## 3rd Qu.:1.40e+09 Arts & Entertainment : 55195   
## Max. :1.40e+09 Professional & Other Places: 52477   
## (Other) : 61434   
## cate\_l2 datetime   
## Gym / Fitness Center: 29733 Min. :2014-02-01 00:00:04   
## Bar : 22697 1st Qu.:2014-03-07 18:23:26   
## Airport : 22678 Median :2014-04-09 17:42:47   
## Office : 22552 Mean :2014-04-09 01:49:43   
## Coffee Shop : 16887 3rd Qu.:2014-05-11 14:30:32   
## American Restaurant : 14160 Max. :2014-06-13 23:58:32   
## (Other) :399658 NA's :69   
## conds winddird windspd temperatur   
## Clear :255476 Min. : 0.0 Min. : 0 Min. :-12.2   
## Overcast :131709 1st Qu.: 0.0 1st Qu.: 6 1st Qu.: 3.3   
## Mostly Cloudy : 42028 Median : 0.0 Median : 9 Median : 11.7   
## Partly Cloudy : 28428 Mean : 96.5 Mean :10 Mean : 10.7   
## Scattered Clouds: 23741 3rd Qu.:220.0 3rd Qu.:13 3rd Qu.: 17.8   
## Light Rain : 18414 Max. :360.0 Max. :35 Max. : 30.0   
## (Other) : 28569 NA's :33147 NA's :188   
## fog rain snow thunder   
## Mode :logical Mode :logical Mode :logical Mode :logical   
## FALSE:524895 FALSE:501401 FALSE:522028 FALSE:528365   
## TRUE :3470 TRUE :26964 TRUE :6337 NA's :0   
## NA's :0 NA's :0 NA's :0   
##   
##   
##   
## tornado hour yearday weekday   
## Mode :logical 19 : 52321 Length:528365 Length:528365   
## FALSE:528365 18 : 47742 Class :character Class :character   
## NA's :0 20 : 41059 Mode :character Mode :character   
## 13 : 35048   
## 17 : 34523   
## 14 : 32393   
## (Other):285279   
## isweekend   
## Saturday: 96134   
## Sunday : 77757   
## Workday :354474   
##   
##   
##   
##

### 1. Temporal analysis

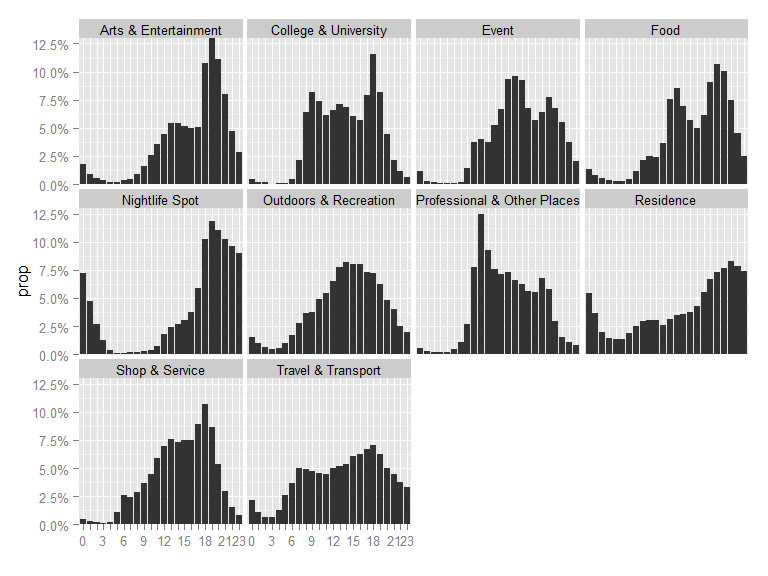
#### Frequency domain

L\_category = split(checkin.global, checkin.global$cate\_l1)  
  
DF\_date\_hour\_category = data.frame()  
temp = lapply(L\_category, function(i){  
 df = stats\_by\_date\_hour(i, category = i[1,"cate\_l1"])  
 DF\_date\_hour\_category <<- rbind(DF\_date\_hour\_category,df)  
})  
  
L\_date\_hour\_category=split(DF\_date\_hour\_category,DF\_date\_hour\_category$cate\_l1)  
## plot  
par(mfrow=c(3,4))  
temp = lapply(seq\_along(L\_date\_hour\_category), function(i){  
 fre = spec.pgram(L\_date\_hour\_category[[i]]$prop, plot=FALSE)  
 plot(fre[["freq"]][1:400], fre[["spec"]][1:400],   
 type="l", main= names(L\_date\_hour\_category[i]),  
 xlab="Frequency", ylab="Spectrum")  
 })  
rm(temp)



#### Probability distribution

DF\_hour\_category = data.frame()  
temp = lapply(L\_category, function(i){  
 df = stats\_checkin\_by\_hour(i, category = i[1,"cate\_l1"])  
 DF\_hour\_category <<- rbind(DF\_hour\_category,df)  
})  
rm(temp)  
## plot  
ggplot(DF\_hour\_category, aes(x=hour,y=prop)) +   
 geom\_bar(stat="identity") +  
 xlab("") +  
 facet\_wrap(~cate\_l1, ncol=4, nrow=3) +  
 coord\_cartesian(ylim = c(0,0.13)) +  
 scale\_y\_continuous(labels = percent) +  
 scale\_x\_discrete(breaks=levels(DF\_hour\_category$hour),   
 labels=c("0","","","3","","","6","","","9","","","12","","","15","","","18","","","21","","23"))

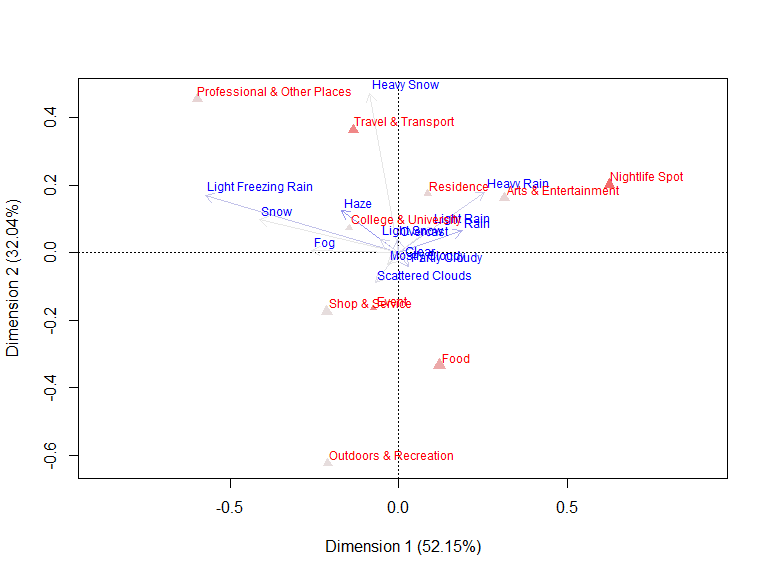


### 2. Meteorological analysis

cate.conds = xtabs(~conds+cate\_l1, data=checkin.global)  
#prop.table(cate.conds, 1) # row percentages  
#prop.table(cate.conds, 2) # column percentages  
fit <- ca(cate.conds)  
#print(fit) # basic results  
summary(fit) # extended results

##   
## Principal inertias (eigenvalues):  
##   
## dim value % cum% scree plot   
## 1 0.003175 52.2 52.2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
## 2 0.001950 32.0 84.2 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*   
## 3 0.000545 8.9 93.1 \*\*\*\*   
## 4 0.000175 2.9 96.0 \*   
## 5 8.4e-050 1.4 97.4 \*   
## 6 6.3e-050 1.0 98.4   
## 7 5.7e-050 0.9 99.4   
## 8 2.7e-050 0.4 99.8   
## 9 1.1e-050 0.2 100.0   
## -------- -----   
## Total: 0.006087 100.0   
##   
##   
## Rows:  
## name mass qlt inr k=1 cor ctr k=2 cor ctr   
## 1 | Cler | 484 588 67 | 12 175 22 | -19 414 86 |  
## 2 | Fog | 2 622 33 | -258 622 39 | 6 0 0 |  
## 3 | Haze | 24 961 184 | -171 629 222 | 124 332 190 |  
## 4 | HvyR | 5 924 84 | 254 615 99 | 180 309 81 |  
## 5 | HvyS | 0 157 5 | -86 5 0 | 472 152 3 |  
## 6 | LgFR | 0 742 32 | -574 682 42 | 169 59 6 |  
## 7 | LghR | 35 952 90 | 97 594 103 | 75 358 101 |  
## 8 | LghS | 8 245 24 | -56 164 7 | 39 81 6 |  
## 9 | MstC | 80 721 42 | -33 342 27 | -35 379 49 |  
## 10 | Ovrc | 249 891 75 | -5 15 2 | 40 876 204 |  
## 11 | PrtC | 54 595 37 | 29 198 14 | -41 397 46 |  
## 12 | Rain | 11 942 74 | 187 841 120 | 65 101 23 |  
## 13 | SctC | 45 886 105 | -68 329 66 | -89 557 183 |  
## 14 | Snow | 4 875 148 | -413 828 235 | 98 47 22 |  
##   
## Columns:  
## name mass qlt inr k=1 cor ctr k=2 cor ctr   
## 1 | ArtE | 104 722 83 | 55 619 98 | 22 103 26 |  
## 2 | CllU | 18 557 24 | -62 485 22 | 24 72 5 |  
## 3 | Evnt | 5 497 24 | -59 124 6 | -102 373 27 |  
## 4 | Food | 294 772 56 | 12 134 14 | -27 638 112 |  
## 5 | NghS | 121 956 225 | 101 898 387 | 26 58 40 |  
## 6 | OtdR | 66 834 177 | -46 131 44 | -107 703 389 |  
## 7 | PrOP | 99 987 256 | -107 727 357 | 64 260 208 |  
## 8 | Rsdn | 27 315 43 | 29 90 7 | 46 225 30 |  
## 9 | ShpS | 157 677 50 | -30 479 46 | -20 198 31 |  
## 10 | TrvT | 108 812 63 | -23 151 18 | 49 662 131 |

#plot(fit) # symmetric map  
plot(fit, mass = TRUE, contrib = "absolute", map ="rowgreen",   
 arrows = c(TRUE, FALSE)) # asymmetric map



### 3. Model - derivation and corresponding functions

Under the assumption is independent from

since is independent from ,

therefore,

$$Exp[P(C=k|H=i,W=j)]=Exp[ \frac{P(H=i,W=j|C=k)\*P(C=k)} {P(H=i)\*P(W=j)}] \\\
=\frac{P(H=i|C=k)\*P(W=j|C=k)\*P(C=k)}{P(H=i)\*P(W=j)} \\\
=\frac{\frac{P(H=i,C=k)}{P(C=k)}\*\frac{P(W=j,C=k)}{P(C=k)}\*P(C=k)}{P(H=i)\*P(W=j)} \\\
=\frac{P(C=k|H=i)\*P(H=i)\*P(C=k|W=j)\*P(W=j)}{P(H=i)\*P(W=j)\*P(C=k)} \\\
=\frac{P(C=k|H=i)\*P(C=k|W=j)}{P(C=k)}$$

checkin.single = read.csv( paste0(basedir, "data\\UserA.csv"),   
 header=TRUE, sep=",", na.strings = "none",  
 colClasses = c("numeric","numeric","factor","factor", "numeric","numeric",  
 "numeric","character","factor","factor")  
)  
checkin.single$datetime = strptime( strtrim(checkin.single$localtime,19), format="%Y-%m-%d %H:%M:%S")  
  
generate.dataframe = function(checkin.global, weather){  
 ## join checkin data with weather data based on timestamps   
 checkin.global = joindfsbytime(checkin.global, weather)  
   
 ## deal with time   
 checkin.global$hour = as.factor(format(checkin.global$datetime,"%H"))  
 checkin.global$yearday = format(checkin.global$datetime,"%j")  
 checkin.global$weekday = format(checkin.global$datetime,"%w")  
 checkin.global$isweekend = as.factor(ifelse( ( checkin.global$weekday>5 ), "Saturday",   
 ifelse( ( checkin.global$weekday<1 ),"Sunday","Workday")))  
   
 ## add record for last checkin  
 # checkin.global = copylastcheckinrec(checkin.global)  
 checkin.global = checkin.global[complete.cases(checkin.global$conds),]  
   
 checkin.global  
}  
  
checkin.single = generate.dataframe(checkin.single, weather)

get.temporal.impact = function(dataframe,hour){  
 dataframe.in.hour = checkin.single[which(checkin.single$hour==hour),]  
 phi.h = nrow(dataframe.in.hour)  
   
 list.category = split(dataframe.in.hour, dataframe.in.hour$cate\_l2)  
 sapply(list.category, function(i){  
 nrow(i)/phi.h  
 })  
}

get.meteorologica.impact = function(fit){  
   
 weather.id = which(fit[["rownames"]]==weather)  
 ref.vec = fit[["rowcoord"]][weather.id,1:2]  
 cate.all = fit[["colcoord"]][,1:2]  
   
 intercepts = apply(cate.all, 1, function(x){   
 (x[1]\*ref.vec[1]+x[2]\*ref.vec[2])/(ref.vec[1]^2+ref.vec[2]^2)   
 } )  
   
 intercepts / sum(intercepts)  
}

get.weather.weight = function(fit){  
   
 weather.all = fit[["rowcoord"]][,1:2]  
   
 mag = apply(weather.all, 1, function(x){   
 sqrt( (x[1]^2+x[2]^2) )  
 } )  
   
 mag / max(mag)  
}  
  
get.weighted.meteorological.impact = function(dataframe, weather){  
 cate.conds = xtabs(~conds+cate\_l2, data=dataframe)  
 fit <- ca(cate.conds)  
   
 unweighted = get.meteorologica.impact(fit)  
 weights = get.weather.weight(fit)  
   
 weather.id = which(fit[["rownames"]]==weather)  
   
 vec = weights[weather.id] \* (unweighted-mean(unweighted)) + mean(unweighted)  
 names(vec) = fit[["colnames"]]  
   
 vec  
   
}

get.denominator = function(dataframe,hour,weather){  
   
 phi.h = nrow(dataframe)  
   
 list.category = split(dataframe, dataframe$cate\_l2)  
 denominator = sapply(list.category, function(i){  
 nrow(i)/phi.h  
 })  
   
 denominator  
  
   
}

get.overall.relevance = function(dataframe,hour, weather){  
   
 p.k = get.denominator(dataframe)  
 p.ki = get.temporal.impact(dataframe, hour)  
 cates.list = names(p.k)  
 p.kj = get.weighted.meteorological.impact(checkin.global, weather)[cates.list]  
   
 p.kij = p.ki \* p.kj / p.k  
}

### Experiment

generate.list = function(dataframe, hour, day,n){  
   
 reference.data = dataframe[which(dataframe$hour==hour & dataframe$yearday == day),]   
 weather = reference.data[1,"conds"]  
   
 places.been.to = unique(reference.data$cate\_l2)  
   
 probs = get.overall.relevance(dataframe, hour, weather)  
 probs = probs[probs>0]  
 places.predicted = names(probs)[order(probs,decreasing=TRUE)]  
   
 list("places.been.to"=places.been.to, "places.predicted" =places.predicted[1:n])  
   
}

### Verification for single user

evaluation.vec = function(gen.list){  
 real = gen.list[["places.been.to"]]  
 pred = gen.list[["places.predicted"]]  
  
 correct = 0  
 for(i in 1:length(pred)){  
 prediction = pred[i]  
 if(length(which(real==prediction)))  
 correct = correct+1  
 }  
   
 real.count = length(real)  
 pred.count = length(pred)  
  
 c("cor"=correct,"real"=real.count,"pred"=pred.count)  
   
}

# random sample  
all.id = 1:nrow(checkin.single)  
sample.id = sample(all.id, 20)  
  
performance = data.frame()  
  
for (i in 1:20){  
 id = sample.id[i]  
 hour = checkin.single[id,"hour"]  
 day = checkin.single[id,"yearday"]  
   
 # prediction list  
 pred.list = generate.list(checkin.single, hour, day, 5)  
 performance=rbind(performance, evaluation.vec(pred.list))  
   
}