IPDS Final Project

Avery DeWindt, Victoria Sharam

Professor Horton

T TH 11:00 AM - 12:15 PM

17 December 2018

1) Using Python, download the main table from this page: <https://en.wikipedia.org/wiki/Farebox_recovery_ratio>

**Code:**

from pandas.io.html import read\_html

import pickle

import pandas as pd

import sqlite3

page = 'https://en.wikipedia.org/wiki/Farebox\_recovery\_ratio'

wikitables = read\_html(page)

table = wikitables[1]

pickle.dump(table, open('wiki\_table.pkl', 'wb'))

table = pickle.load(open('wiki\_table.pkl', 'rb'))

continents = table[0][1:]

countries = table[1][1:]

systems = table[2][1:]

ratios = table[3][1:]

faresystems = table[4][1:]

farerates = table[5][1:]

years = table[6][1:]

2) Write Python code to "clean up" this dataset.

* All numbers are only numbers
* All currencies are converted to USD
* All fractions as floats
* Make a choice how to represent fares and explain your approach

**Code:**

# Exchange Rates collected from websites below

# https://www.wellsfargo.com/foreign-exchange/currency-rates/?fbclid=IwAR1BbsB4bzKsF7LtnGwFE4yv97mxISK-KNiK7oIAjIeHtfRUsRH\_alhjDto

# Netherlands: https://www.google.com/search?q=exchange+rate+netherlands&oq=exchange+rate+netherlands&aqs=chrome..69i57j0l5.11327j1j4&sourceid=chrome&ie=UTF-8

# Germany: https://www.google.com/search?ei=GwgXXK6UDoeGtQXwy62ACg&q=exchange+rate+germany&oq=exchange+rate+germany&gs\_l=psy-ab.3..0i70i258j0l4j0i22i30l5.20274.21738..22300...0.0..0.373.1283.2j3j1j1......0....1..gws-wiz.......0i71.GtrtuiITX5Q

# Belgium: https://www.google.com/search?ei=MggXXK7cMYLwsAW42YagBw&q=exchange+rate+belgium&oq=exchange+rate+belgium&gs\_l=psy-ab.3..0i70i258j0j0i22i30l8.6543.7446..7612...0.0..1.362.1263.0j4j2j1......0....1..gws-wiz.......0i71.ucKIwstMbIs

# UK: https://www.google.com/search?ei=OwgXXPeqKoHKswWtibugCg&q=exchange+rate+uk&oq=exchange+rate+uk&gs\_l=psy-ab.3..0i71l8.6916.7007..7144...0.0..0.0.0.......0....1..gws-wiz.trANGYP-Ycs

# Spain: https://www.google.com/search?ei=QwgXXKL7NMyXtgXb\_oywBw&q=exchange+rate+spain&oq=exchange+rate+spain&gs\_l=psy-ab.3..0i70i258j0l3j0i22i30l6.13946.15055..15321...1.0..0.300.819.2j3j0j1......0....1..gws-wiz.......0i71j0i67.kVERwSNA0Jw

# Italy: https://www.google.com/search?ei=jwgXXLmJM4nIsQWlzZ\_QAg&q=exchange+rate+italy&oq=exchange+rate+italy&gs\_l=psy-ab.3..0i70i258j0l2j0i22i30l7.7599.8080..8278...0.0..1.391.978.1j2j1j1......0....1..gws-wiz.......0i71.4uYJ0VlVoV4

# France: https://www.google.com/search?ei=qAgXXIjKA4iMsQXj-KngAQ&q=exchange+rate+France&oq=exchange+rate+France&gs\_l=psy-ab.3..0l2j0i10l8.17025.17025..17272...0.0..0.113.113.0j1......0....1j2..gws-wiz.......0i71.w5wh9LhIY\_Q

# Austria: https://www.google.com/search?ei=uggXXLXaGI-osgXiwJGQAQ&q=exchange+rate+Austria&oq=exchange+rate+Austria&gs\_l=psy-ab.3..0l2j0i22i30l2j0i22i10i30j0i22i30l4j0i22i10i30.32283.32283..32473...0.0..0.120.120.0j1......0....1j2..gws-wiz.......0i71.ZUUeNrnf6fg

# Finland: https://www.google.com/search?ei=3AgXXPGUAo6ItQXDn4qoDQ&q=exchange+rate+Finland&oq=exchange+rate+Finland&gs\_l=psy-ab.3..0l2j0i22i30l8.33372.33372..33686...0.0..0.248.248.2-1......0....1j2..gws-wiz.......0i71.cpOJdKTxg2g

ConversionRates = {}

for country in countries:

ConversionRates[country]=0

if country=='Hong Kong':

ConversionRates[country]=0.136182

elif country=='Japan':

ConversionRates[country]=0.0092902

elif country=='Pakistan':

ConversionRates[country]=0.007948

elif country=='Taiwan':

ConversionRates[country]=0.035989

elif country=='Singapore':

ConversionRates[country]=0.774833

elif country=='China':

ConversionRates[country]=0.1552024

elif country=='Netherlands':

ConversionRates[country]=1.13

elif country=='Germany':

ConversionRates[country]=1.13

elif country=='Belgium':

ConversionRates[country]=1.13

elif country=='Denmark':

ConversionRates[country]=0.160436

elif country=='UK':

ConversionRates[country]=1.26

elif country=='Spain':

ConversionRates[country]=1.13

elif country=='Italy':

ConversionRates[country]=1.13

elif country=='Czech Republic':

ConversionRates[country]=0.046953

elif country=='France':

ConversionRates[country]=1.13

elif country=='Sweden':

ConversionRates[country]=0.116198

elif country=='Austria':

ConversionRates[country]=1.13

elif country=='Finland':

ConversionRates[country]=1.13

elif country=='Switzerland':

ConversionRates[country]=1.058201

elif country=='US':

ConversionRates[country]=1

elif country=='Canada':

ConversionRates[country]=0.786968

elif country=='New Zealand':

ConversionRates[country]=0.7152

elif country=='Australia':

ConversionRates[country]=0.756

def CleanRatio(raw\_ratio):

a = raw\_ratio.split("%")

a = float(a[0])/100.00

return "%.2f" % a

def CleanYear(raw\_year):

f = raw\_year.split("[")

return f[0]

# Fare systems categorized as either Zone-, Distance-, Flatrate-based, or Other.

def CleanFaresystem(raw\_faresystem):

s = str(raw\_faresystem)

s = s.split()

long\_string = ""

for chr in s:

long\_string += chr

if long\_string=='nan':

long\_string=""

else:

long\_string=long\_string.lower()

if long\_string[0:4]=='zone':

long\_string='zone\_based'

elif long\_string[0:4]=='dist':

long\_string='distance\_based'

elif long\_string[0:4]=='flat':

long\_string='flat\_rate'

else:

long\_string='other'

return long\_string

# Fare rates are represented by the lower bound, if range was given, or by first option presented in cell.

accaptable\_characters = ['1','2','3','4','5','6','7','8','9','0','.','+','(','-','t']

unacceptable\_characters = ['+', '(', '-','t']

def CleanFarerateStepOne(raw\_farerate):

d = str(raw\_farerate)

d = d.split()

long\_string = ""

for chr in d:

long\_string += chr

if long\_string=='nan':

return ""

else:

return long\_string

def CleanFarerateStepTwo(raw\_farerate):

long\_string = ""

for chr in raw\_farerate:

if chr in accaptable\_characters:

long\_string += chr

return long\_string

def CleanFarerateStepThree(i):

to\_edit = []

for chr in i:

if chr in unacceptable\_characters:

to\_edit.append(i)

for i in to\_edit:

for chr in i:

if chr=='+':

i = i.split('+')

return i[0]

break

elif chr=='(':

i = i.split('(')

return i[0]

break

elif chr=='-':

i = i.split('-')

return i[0]

break

elif chr=='t':

i = i.split('t')

return i[0]

return i

def ConvertUSD(pos, clean\_farerate):

if clean\_farerate > float(0):

for r in range(len(rates)):

for c in range(len(countries\_clean)):

for country in ConversionRates.keys():

if countries\_clean[c]==country and pos==c and c==r:

exchange\_rate = ConversionRates[str(country)]

converted = clean\_farerate \* exchange\_rate

return "%.2f" % converted

break

elif clean\_farerate==float(0):

return "%.2f" % clean\_farerate

clean\_ratios = []

for ratio in ratios:

clean\_ratios.append(CleanRatio(ratio))

clean\_years = []

for year in years:

clean\_years.append(CleanYear(year))

clean\_faresystems = []

for faresystem in faresystems:

clean\_faresystems.append(CleanFaresystem(faresystem))

clean\_farerates\_s1 = []

for farerate in farerates:

clean\_farerates\_s1.append(CleanFarerateStepOne(farerate))

clean\_farerates\_s2 = []

for farerate in clean\_farerates\_s1:

clean\_farerates\_s2.append(CleanFarerateStepTwo(farerate))

clean\_farerates = []

for farerate in clean\_farerates\_s2:

clean\_farerates.append(CleanFarerateStepThree(farerate))

rates = []

for i in clean\_farerates:

if i=='':

rates.append(float(0))

else:

rates.append(float(i))

countries\_clean = []

for i in countries:

countries\_clean.append(str(i))

continents\_clean = []

for i in continents:

continents\_clean.append(str(i))

systems\_clean = []

for i in systems:

systems\_clean.append(str(i))

USD\_FareRates = []

for pos in range(len(rates)):

USD\_FareRates.append(ConvertUSD(pos, rates[pos]))

df=pd.DataFrame(continents\_clean,columns=['Continent'])

df['Country']=countries\_clean

df['System']=systems\_clean

df['Ratio']=clean\_ratios

df['Fare\_system']=clean\_faresystems

df['Fare\_rate\_USD']=USD\_FareRates

df['Year']=clean\_years

3) Write the cleaned up dataset to a local SQLite database.

Code:

db\_file = "Farebox.db"

conn = sqlite3.connect(db\_file)

cur = conn.cursor()

create\_table\_sql = """CREATE TABLE IF NOT EXISTS FareBox (

continents VARCHAR(30),

countries VARCHAR(30),

systems VARCHAR(30) PRIMARY KEY,

ratios FLOAT,

faresystems VARCHAR(30),

farerates FLOAT,

years INTEGER

); """

cur.execute(create\_table\_sql)

for r in range(len(df['Continent'])):

continent=df['Continent'][r]

country=df['Country'][r]

system=df['System'][r]

ratio=df['Ratio'][r]

faresystem=df['Fare\_system'][r]

farerate=df['Fare\_rate\_USD'][r]

year=df['Year'][r]

sql\_code = """ INSERT OR REPLACE INTO FareBox VALUES

('%s', '%s', '%s', '%s', '%s', '%s', '%s')""" % (

continent, country, system, ratio, faresystem, farerate, year)

cur.execute(sql\_code)

conn.commit()

4) Access the SQLite database from R, and create the following plots:

* The unconditioned distribution of farebox ratio (called "Ratio" in the Wiki table)
* Scatter plot showing farebox ratio versus fare rates (i.e., what it costs to ride) for flat rate systems
* Create a facet plot of the distribution of farebox ratios, by fare system i.e., (facet\_wrap(~fare\_system, ncol = ...).
* Create a faceted plot showing the distribution of farebox ratios by continent (use a histogram or density)

**Code:**

rm(list=ls())

setwd("C:/Users/Victoria Sharam/sql\_practice/FinalProject")

install.packages("RSQLite")

install.packages("ggplot2")

library(RSQLite)

library(ggplot2)

db <- dbConnect(SQLite(), dbname="Farebox.db")

dbListTables(db)

dbListFields(db, "Farebox")

all\_data <- dbGetQuery(db,"SELECT \* FROM Farebox")

# Q1: The unconditioned distribution of farebox ratio (called "Ratio" in the Wiki table)

ggplot(data=all\_data, aes(x=ratios)) +

geom\_density()

# Q2: Scatter plot showing farebox ratio versus fare rates (i.e., what it costs to ride) for flat rate systems

plot2data <- dbGetQuery(db,"SELECT \* FROM FareBox WHERE faresystems ='flat\_rate' AND farerates > 0")

ggplot(data=plot2data, aes(x=ratios, y=farerates)) +

geom\_point()+

geom\_smooth()

# Q3: Create a facet plot of the distribution of farebox ratios, by fare system i.e.,

# (facet\_wrap(~fare\_system, ncol = ...).

ggplot(data=all\_data, aes(x=ratios)) +

geom\_density() +

facet\_wrap(~faresystems, ncol=1)

# Q4:Create a faceted plot showing the distribution of farebox ratios by

# continent (use a histogram or density)

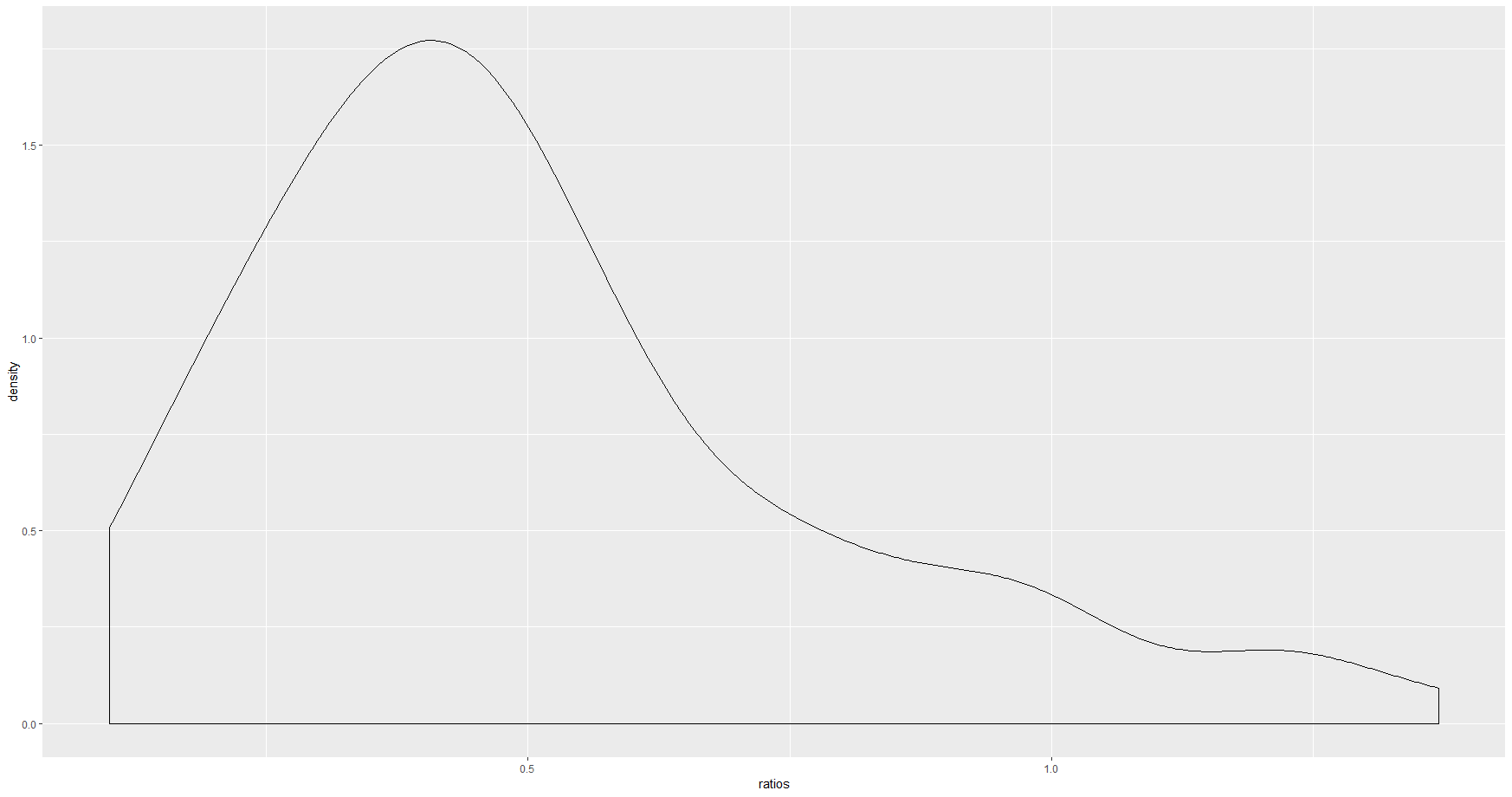
ggplot(data=all\_data, aes(x=ratios, fill=continents)) +

geom\_histogram() +

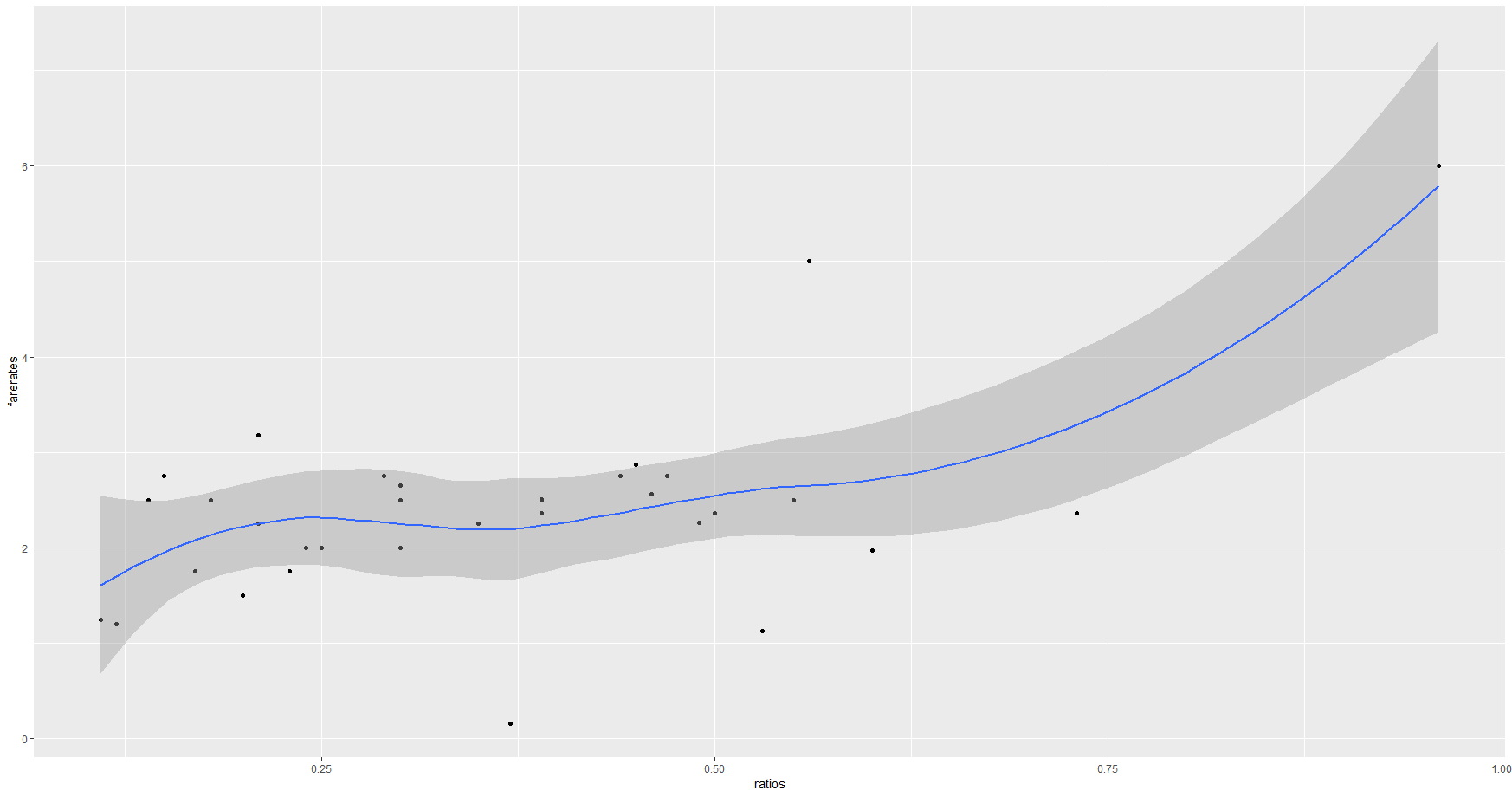
facet\_wrap(~continents, ncol=1)

**Output:**

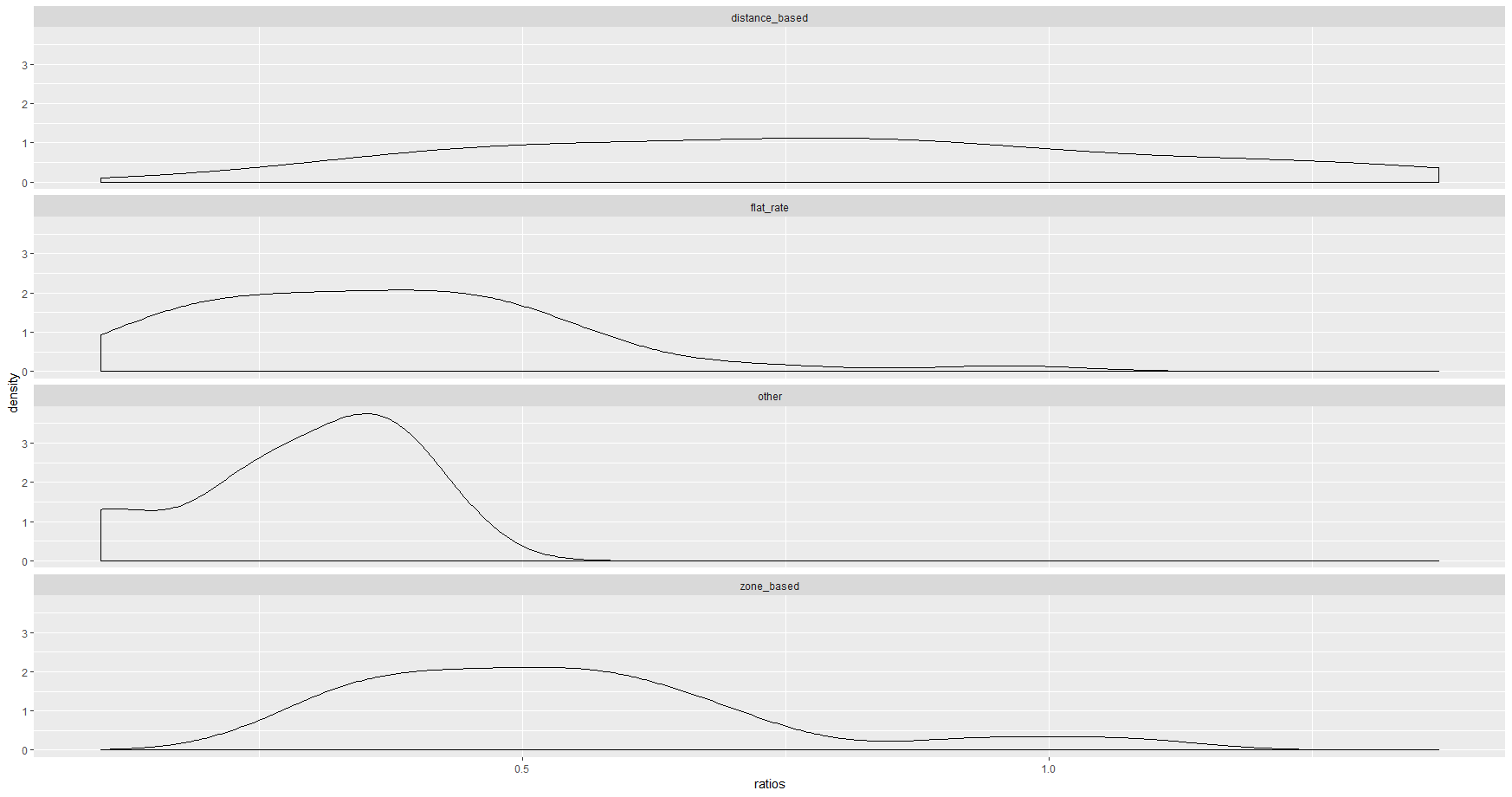
* The unconditioned distribution of farebox ratio (called "Ratio" in the Wiki table)



* Scatter plot showing farebox ratio versus fare rates (i.e., what it costs to ride) for flat rate systems



* Create a facet plot of the distribution of farebox ratios, by fare system i.e., (facet\_wrap(~fare\_system, ncol = ...).



* Create a faceted plot showing the distribution of farebox ratios by continent (use a histogram or density)



5) Run a prediction model of the fare collection ratio on predictors---what do we learn?

Code:

# Q5: Run a prediction model of the fare collection ratio on predictors---

# what do we learn?

plot5data = dbGetQuery(db,"SELECT \* FROM FareBox WHERE farerates>0")

PredictionModelOne = lm(ratios~faresystems+countries+farerates+years, data=plot5data)

summary(PredictionModelOne)

PredictionModelTwo = lm(ratios~faresystems+continents+farerates+years, data=plot5data)

summary(PredictionModelTwo)

PredictionModelThree = lm(ratios~faresystems+farerates+years, data=plot5data)

summary(PredictionModelThree)

**Output:**

Prediction Model One explains most of the variation in the data (i.e. has the highest R-Squared Adj.). The very low p value of 2.26e-11 tells us the model is statistically significant and the f statistic of 11.56 attests to that. The R-Squared of 89.5% indicates the model is a strong predictor of fare ratios. The standard error .1216 tells us the multiplicative factor of the confidence interval is e±.2432 with true fare ratios falling between 78.5% - 127.5%.