MSAN 593 - HW5

Andre Guimaraes Duarte

August 16, 2016

```r
# Check if ggplot is installed and install/load the package
if(!"ggplot2" %in% rownames(installed.packages())){
  install.packages("ggplot2", dependencies = T)
}
library(ggplot2)

# Check if lubridate is installed and install/load the package
if(!"lubridate" %in% rownames(installed.packages())){
  install.packages("lubridate", dependencies = T)
}
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

```r
# Check if magrittr is installed and install/load the package
if(!"magrittr" %in% rownames(installed.packages())){
  install.packages("magrittr", dependencies = T)
}
library(magrittr)

# Check if corrplot is installed and install/load the package
if(!"corrplot" %in% rownames(installed.packages())){
  install.packages("corrplot", dependencies = T)
}
library(corrplot)

# Check if tidyr is installed and install/load the package
if(!"tidyr" %in% rownames(installed.packages())){
  install.packages("tidyr", dependencies = T)
}
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```r
# Check if scales is installed and install/load the package
if(!"scales" %in% rownames(installed.packages())){
  install.packages("scales", dependencies = T)
}
library(scales)

# Check if reshape2 is installed and install/load the package
if(!"reshape2" %in% rownames(installed.packages())){
  install.packages("reshape2", dependencies = T)
}
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following object is masked from 'package:tidyr':
##
##     smiths
```

Question 1

All plots for this question are on a separate `plots.pdf` file.

In this exercise, we are given a data set from a hotel, which we load into a variable `hotel`.

Task 1

Our first task is to clean and validate the data. We first take a look at its structure using `str`.

```r
hotel <- read.csv("hotelData.csv", header = T, stringsAsFactors = F, na.strings = "")
str(hotel, strict.width = "w")
```

```
## 'data.frame':    45360 obs. of  8 variables:
## $ room : chr "A" "A" "A" "A" ...
## $ date : chr "2009-01-01 00:00:00" "2009-01-02 00:00:00" "2009-01-03
##    00:00:00" "2009-01-04 00:00:00" ...
## $ area : int 988 988 988 988 988 988 988 988 988 988 ...
## $ event_min: int 0 0 0 0 0 0 120 0 0 0 ...
## $ max_min : int 1080 1080 1080 1080 1080 1080 1080 1080 1080 1080 ...
## $ year : int 2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
## $ month : chr "January" "January" "January" "January" ...
## $ day : int 1 2 3 4 5 6 7 8 9 10 ...
```

We can see that we have 45360 rows and 8 columns with relvant information. From the structure, we can see that the rooms are names with capital letters, from `"A"` to `"U"`, and they were imported as character. We will transform those into an ordered factor from `"A"` to `"U"`.

In addition, the dates have also been imported as character. They are in the format `%Y-%m-%d %H:%M:%S`, but the hours, minutes, and seconds are always `0`. So we will convert them into `Date`, while only keeping the year, month, and day.

The months have been imported as character. They can be converted to an ordered factor from `"January"` to `"December"`.

```r
hotel$room <- factor(hotel$room, levels = unique(hotel$room), ordered = T)
hotel$month <- factor(hotel$month, levels = month.name, ordered = T)
hotel$date <- as.Date(hotel$date, "%Y-%m-%d")
str(hotel, strict.width = "w")
```

```
## 'data.frame':    45360 obs. of  8 variables:
## $ room : Ord.factor w/ 21 levels "A"<"B"<"C"<"D"<..: 1 1 1 1 1 1 1 1 1 1
##    ...
## $ date : Date, format: "2009-01-01" "2009-01-02" ...
## $ area : int 988 988 988 988 988 988 988 988 988 988 ...
## $ event_min: int 0 0 0 0 0 0 120 0 0 0 ...
## $ max_min : int 1080 1080 1080 1080 1080 1080 1080 1080 1080 1080 ...
## $ year : int 2009 2009 2009 2009 2009 2009 2009 2009 2009 2009 ...
## $ month : Ord.factor w/ 12 levels "January"<"February"<..: 1 1 1 1 1 1 1 1
##    1 1 ...
## $ day : int 1 2 3 4 5 6 7 8 9 10 ...
```

This is better. Now let's see if we have NAs in our data. We run the command `any(is.na(hotel))`, which returns `TRUE` if it finds one `NA` entry. Running this command returns FALSE. So our data is complete.

Another check we can perform is to make sure that `event_min` is lower than or equal to `max_min`. For any entry where this condition is violated, we will set `event_min` to `max_min`.

```
hotel[hotel$event_min > hotel$max_min,]$event_min <- hotel$max_min[1]
```

The data seems good now. We have not had to remove any entry in the data set, although we had to change the `event_min` for some rows.

Task 2

We now wish to create a line graph plotting the room utilization rate in percent (y) against the day of the year from 1 to 365 (x), horizontally faceted by year (six facets). We use `ggplot` for this. We also use the package `lubridate` to better work with dates. We create one graph per room, so 21 graphs.

```
hotel$dayOfYear <- yday(hotel$date)
hotel$utilization <- hotel$event_min/hotel$max_min

for(i in seq_along(levels(hotel$room))){
  hotel[hotel$room == levels(hotel$room)[i],] %>%
    ggplot(aes(dayOfYear, utilization, colour = factor(year))) +
    geom_line(size = .25) +
    facet_grid(year ~ .) +
    xlab("\n Day of Year") +
    ylab("Utilization\n") +
    ggtitle(paste("Room '", levels(hotel$room)[i], "' Utilization Rates\n",
                  sep = "")) +
    theme(legend.position = "none") +
    theme(axis.text = element_text(size=12),
          strip.text.y = element_text(size = 18),
          title = element_text(size = 20)) +
    scale_x_continuous(breaks = seq(0, 365, 30)) +
    scale_y_continuous(labels = percent)
  #ggsave(paste0("q1t2-", i, ".pdf"), height = 8.5, width = 11)
}
```

Task 3

We wish now to create a line graph plotting the room utilization rate in percent (y) against the day of the year from 1 to 365 (x), horizontally faceted by weekday (seven facets, Monday to Sunday). We create one graph per room, so 21 graphs.

```
hotel$dayOfWeek <- factor(weekdays(hotel$date), levels = c("Monday",
                                                           "Tuesday",
                                                           "Wednesday",
                                                           "Thursday",
                                                           "Friday",
                                                           "Saturday",
                                                           "Sunday"), ordered = T)
for(i in seq_along(levels(hotel$room))){
```

```r
  hotel[hotel$room == levels(hotel$room)[i],] %>%
    ggplot(aes(dayOfYear, utilization, colour = dayOfWeek)) +
    geom_line(size = .25) +
    facet_grid(dayOfWeek ~ .) +
    xlab("\n Day of Year") +
    ylab("Utilization\n") +
    ggtitle(paste("Room '", levels(hotel$room)[i],
                  "' Utilization Rates by Day of the Week\n", sep = "")) +
    theme(legend.position = "none") +
    theme(axis.text = element_text(size=12),
          strip.text.y = element_text(size = 12),
          title = element_text(size = 20)) +
    scale_x_continuous(breaks = seq(0, 365, 30)) +
    scale_y_continuous(labels = percent)
  #ggsave(paste0("q1t3-", i, ".pdf"), height = 8.5, width = 11)
}
```

Task 4

We wish to create a line graph plotting the mean room utilization rate in percent (y), against day of the year from 1 to 365 (x), faceted horizontally by either weekday or weekend (two facets). Furthermore, we want to include a semi-opaque area indicating the min and max utilization for any given day. This generates 21 graphs, one for each room.

```r
hotel$weekdayOrWeekend <- factor(ifelse(hotel$dayOfWeek < "Saturday", "Weekday", "Weekend"))

for(i in seq_along(levels(hotel$room))){
  df <- subset(hotel[hotel$room == levels(hotel$room)[i],])
  minUtilizationPerDay <- aggregate(utilization ~ dayOfYear + weekdayOrWeekend, df, min)
  names(minUtilizationPerDay)[3] <- "minimum"
  maxUtilizationPerDay <- aggregate(utilization ~ dayOfYear + weekdayOrWeekend, df, max)
  names(maxUtilizationPerDay)[3] <- "maximum"
  df <- merge(df, minUtilizationPerDay, by = c("dayOfYear", "weekdayOrWeekend"))
  df <- merge(df, maxUtilizationPerDay, by = c("dayOfYear", "weekdayOrWeekend"))

  df %>%
    ggplot(aes(dayOfYear, utilization)) +
      geom_line(stat = "summary", fun.y ="mean", size = .3) +
      geom_ribbon(aes(ymin = minimum, ymax = maximum), alpha = 0.3) +
      facet_grid(weekdayOrWeekend ~ .) +
      xlab("\n Day of Year") +
      ylab("Min, Mean, & Max Utilization\n") +
      ggtitle(paste("Room '", levels(hotel$room)[i], "' Utilization Rates\n", sep = "")) +
      theme(legend.position = "none") +
      theme(axis.text = element_text(size=12),
            strip.text.y = element_text(size = 18),
            title = element_text(size = 20)) +
      scale_x_continuous(breaks = seq(0, 365, 30)) +
      scale_y_continuous(labels = percent)
  #ggsave(paste0("q1t4-", i, ".pdf"), height = 8.5, width = 11)
}
```

Task 5

We now want to create a line graph plotting the mean room utilization rate in percent (y), against day of year from 1 to 365 (x). Furthermore, we will include a semi-opaque area indicating the min and max utilization for any given day. This will generate 21 graphs, one for each room.

```r
for(i in seq_along(levels(hotel$room))){
  df <- subset(hotel[hotel$room == levels(hotel$room)[i],])
  minUtilizationPerDay <- aggregate(utilization ~ dayOfYear, df, min)
  names(minUtilizationPerDay)[2] <- "minimum"
  maxUtilizationPerDay <- aggregate(utilization ~ dayOfYear, df, max)
  names(maxUtilizationPerDay)[2] <- "maximum"
  df <- merge(df, minUtilizationPerDay, by = "dayOfYear")
  df <- merge(df, maxUtilizationPerDay, by = "dayOfYear")

  df %>%
    ggplot(aes(dayOfYear, utilization)) +
      geom_line(stat = "summary", fun.y ="mean", size = .3) +
      geom_ribbon(aes(ymin = minimum, ymax = maximum), alpha = 0.3) +
      xlab("\n Day of Year") +
      ylab("Min, Mean, & Max Utilization\n") +
      ggtitle(paste("Room '", levels(hotel$room)[i], "' Utilization Rates\n",
                    sep = "")) +
      theme(legend.position = "none") +
      theme(axis.text = element_text(size=12),
            strip.text.y = element_text(size = 18),
            title = element_text(size = 20)) +
      scale_x_continuous(breaks = seq(0, 365, 30)) +
      scale_y_continuous(labels = percent)
  #ggsave(paste0("q1t5-", i, ".pdf"), height = 8.5, width = 11)
}
```

Task 6

For each day of the week (Monday - Sunday), we run `kmeans()` on the mean utilizations of each room for each day of the week, with $k \in K = 2, ..., 10$. For each $k \in K$, we create a horizontal Cleveland dot plot plotting room name (y) on mean utilization in percent (x), grouping clusters by color. This will generate Cleveland dot plots per day of the week, and 63 in total. For each weekday, we also create a summary line and dot plot plotting total within SS (y) on number of clusters (x). This will generate an additional seven plots.

```r
meanUtilizationsDayRoom <- aggregate(utilization ~ dayOfWeek + room, hotel, mean)

for(i in 1:7){
  df <- meanUtilizationsDayRoom[meanUtilizationsDayRoom$dayOfWeek == levels(
                                hotel$dayOfWeek)[i],]
  df <- df[order(df$utilization, decreasing = T),]
  SS <- vector("numeric", 9)
  for(k in 2:10){
    kmean <- kmeans(df[, 3], k)
    df$cluster <- factor(kmean$cluster)
    SS[k-1] <- kmean$tot.withinss
    df %>%
      ggplot(aes(utilization, reorder(room, utilization), colour = cluster)) +
        geom_point(size = 3) +
        geom_segment(aes(x = 0, xend = utilization, y = room, yend = room),
                     colour = "black", size = .3) +
        geom_point(size = 3) +
        xlab("\n Mean Utilization") +
        ylab("Room\n") +
        ggtitle(paste(levels(hotel$dayOfWeek)[i], ": ",
                      k, " - Means Grouping by Mean Utilization\n",
                      sep = "")) +
        theme(legend.position = "none",
              axis.text = element_text(size=12),
              strip.text.y = element_text(size = 18),
              title = element_text(size = 18)) +
        scale_y_discrete(labels = paste("Room", df$room)) +
        scale_x_continuous(breaks = seq(0, 1, .05),
                           labels = percent) +
        coord_cartesian(xlim = c(min(df$utilization), max(df$utilization)))
    #ggsave(paste0("q1t6-", i, "-", k, ".pdf"), height = 8.5, width = 11)
  }

  ggplot(NULL, aes(2:10, SS)) +
    geom_line(size = .25) +
    geom_point(size = 2) +
    xlab("\nNumber of Clusters") +
    ylab("Total Within SS\n") +
    ggtitle(paste(levels(hotel$dayOfWeek)[i],
                  ": ",
  "Total Within SS versus Number of Clusters for K-Means Grouping by Mean Utilization\n",
                  sep = "")) +
    theme(legend.position = "none") +
    theme(axis.text = element_text(size=12),
          strip.text.y = element_text(size = 18),
          title = element_text(size = 13)) +
    scale_x_continuous(breaks = 2:10, labels = as.character(2:10))
  #ggsave(paste0("q1t6-", i, ".pdf"), height = 8.5, width = 11)
}
```

Task 7

Using the corrplot package, we create a correlation matrix for mean utilizations for each room for each year. We also include the option where insignificant correlations are crossed out, resulting in six correlation matrices.

```
for(i in unique(hotel$year)){
  pdf(paste0("q1t7-", i, ".pdf"), paper = "letter")
    d <- hotel[hotel$year == i,]
    d <- data.frame(split(d$utilization, d$room))
    names(d) <- paste("Room '", LETTERS[1:21], "'", sep = "")

    M <- cor(d)
    corrplot(M, p.mat = 1-M, method = "square", type = "lower",
           sig.level = .9, insig = "pch", tl.cex = .75, tl.col = "black",
           pch.cex = 3)
    title(main = paste("Correlation Matrix of Utilization by Year:", i, "\n"))
  dev.off()
}
```

Task 8

Using the `corrplot` package, we now create a correlation matrix for mean utilizations for each room for each day of the week. We also include the option where insignificant correlations are crossed out, resulting in seven correlation matrices.

```
for(i in 1:7){
  pdf(paste0("q1t8-", i, ".pdf"), paper = "letter")
    d <- hotel[hotel$dayOfWeek == levels(hotel$dayOfWeek)[i],]
    d <- data.frame(split(d$utilization, d$room))
    names(d) <- paste("Room '", LETTERS[1:21], "'", sep = "")

    M <- cor(d)
    corrplot(M, p.mat = 1-M, method = "square", type = "lower",
             sig.level = .9, insig = "pch", tl.cex = .75, tl.col = "black",
             pch.cex = 3)
    title(main = paste("Correlation Matrix of Utilization by Day:",
                       levels(hotel$dayOfWeek)[i], "\n"))
  dev.off()
}
```

Task 9

Using the `corrplot` package, we finally create a correlation matrix for mean room utilizations. We include the option where insignificant correlations are crossed out. This will result in a single correlation matrix.

```
pdf("q1t9.pdf", paper = "letter")
  d <- data.frame(split(hotel$utilization, hotel$room))
  names(d) <- paste("Room '", LETTERS[1:21], "'", sep = "")

  M <- cor(d)
  corrplot(M, p.mat = 1-M, method = "square", type = "lower",
           sig.level = .95, insig = "pch", tl.cex = .75, tl.col = "black",
           pch.cex = 3)
  title(main = "Correlation Matrix of Utilization")
dev.off()
```

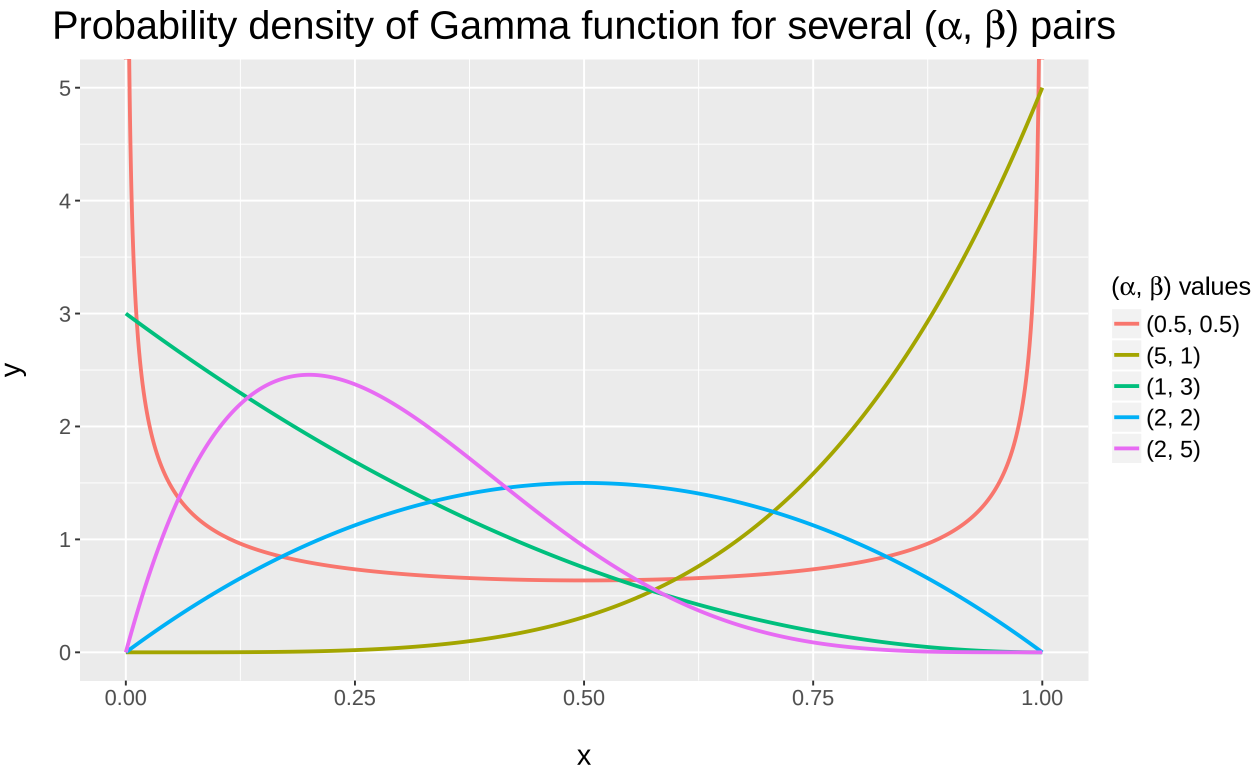
```
## pdf
##   2
```

Question 2

1

Using a single line graph, we generate the probability density functions of the beta distribution where $\alpha, \beta \in 0.5, 0.5, 5, 1, 1, 3, 2, 2, 2, 5$. The probability density function of each α, β tuple is differentiated using color, and a legend is included.

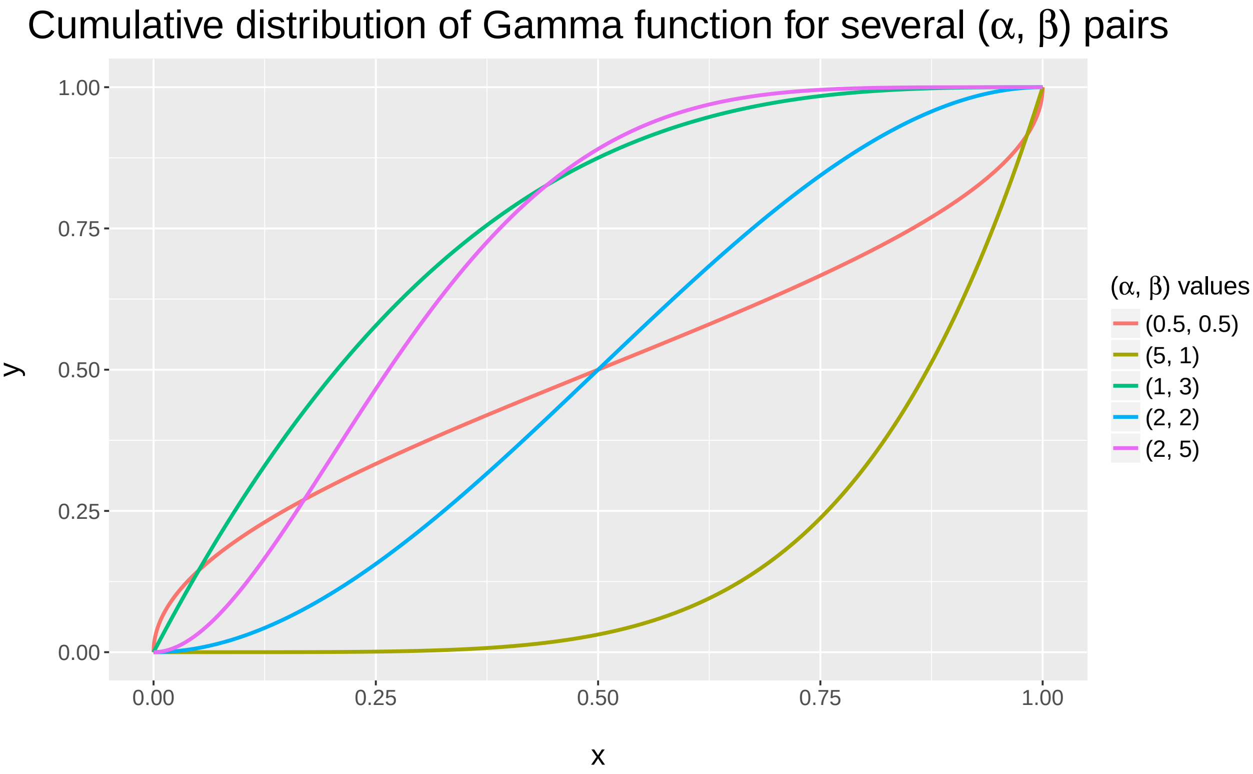
```r
alphas <- c(0.5, 5, 1, 2, 2)
betas <- c(0.5, 1, 3, 2, 5)
x <- seq(0, 1, length.out = 5000)

pdf <- data.frame(
  "x" = x,
  "y1" = dbeta(x, alphas[1], betas[1]),
  "y2" = dbeta(x, alphas[2], betas[2]),
  "y3" = dbeta(x, alphas[3], betas[3]),
  "y4" = dbeta(x, alphas[4], betas[4]),
  "y5" = dbeta(x, alphas[5], betas[5])
)

pdf <- gather(pdf, tuple, y, y1, y2, y3, y4, y5)
pdf$tuple <- as.factor(pdf$tuple)


fig <- ggplot(pdf, aes(x, y, colour = tuple)) +
  geom_line(size = 1) +
  coord_cartesian(ylim = c(0, 5)) +
  ylab("y\n") +
  xlab("\nx") +
  theme(legend.title = element_text(size = 14),
        legend.text = element_text(size = 13),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 16),
        title = element_text(size = 18)) +
  scale_color_manual(expression(paste("(", alpha, ", ", beta, ") values",
                                      sep = "")),
                     labels=paste("(", alphas, ", ", betas, ")", sep = ""),
                     values = c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6",
                                "#E76BF3")) +
  ggtitle(expression(paste("Probability density of Gamma function for several (",
                           alpha, ", ", beta, ") pairs", sep = "")))
#ggsave(plot = fig, filename = paste0("q2t1.pdf"), height = 8.5, width = 11)
print(fig)
```



2

Using a single line graph, we now generate the cumulative distribution functions of the beta distribution where $\alpha, \beta \in 0.5, 0.5, 5, 1, 1, 3, 2, 2, 2, 5$. The cumulative distribution function of each each α, β tuple is differentiated using color, and a legend should be included.

```
cdf <- data.frame(
  "x" = x,
  "y1" = pbeta(x, alphas[1], betas[1]),
  "y2" = pbeta(x, alphas[2], betas[2]),
  "y3" = pbeta(x, alphas[3], betas[3]),
  "y4" = pbeta(x, alphas[4], betas[4]),
  "y5" = pbeta(x, alphas[5], betas[5])
)

cdf <- gather(cdf, tuple, y, y1, y2, y3, y4, y5)
cdf$tuple <- as.factor(cdf$tuple)

fig <- ggplot(cdf, aes(x, y, colour = tuple)) +
  geom_line(size = 1) +
  ylab("y\n") +
  xlab("\nx") +
  theme(legend.title = element_text(size = 14),
        legend.text = element_text(size = 13),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 16),
        title = element_text(size = 18)) +
  scale_color_manual(expression(paste("(", alpha, ", ", beta, ") values",
                                       sep = "")),
                        labels=paste("(", alphas, ", ", betas, ")", sep = ""),
                        values = c("#F8766D", "#A3A500", "#00BF7D", "#00B0F6",
                                   "#E76BF3")) +
  ggtitle(expression(paste("Cumulative distribution of Gamma function for several (",
                          alpha, ", ", beta, ") pairs", sep = "")))
#ggsave(plot = fig, filename = paste0("q2t2.pdf"), height = 8.5, width = 11)
print(fig)
```



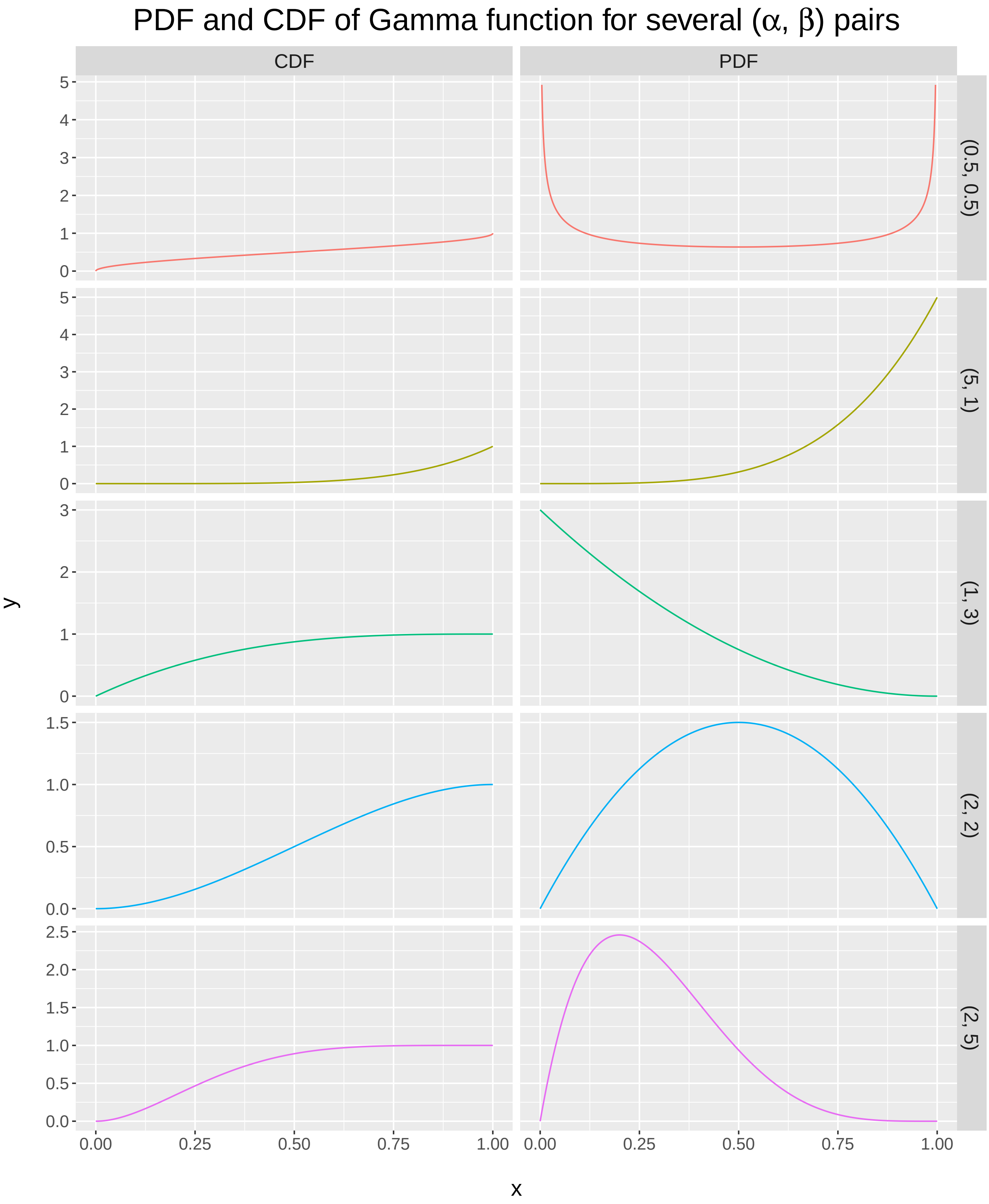
3

Combining output from 2.1 and 2.2, we create a single graphical output with all pdfs and cdfs. They are all line graphs and are faceted using two columns and five rows. The title of each facet column are `pdf` and `cdf`. The titles of the row facets, located on the right hand side of the graph, have the α, β tuple values, i.e., $\alpha, \beta = x, y$, where x and y are the tuple values. The titles have the correct greek symbols α and β, and the tuple values are generated dynamically based on their values (not hard coded). The output is a single graphical object with 10 sub-graphs in two columns and five rows.

```
tuple_names <- list(
  y1 = eval(expression(paste("(", alphas[1], ", ", betas[1], ")", sep = ""))),
  y2 = eval(expression(paste("(", alphas[2], ", ", betas[2], ")", sep = ""))),
  y3 = eval(expression(paste("(", alphas[3], ", ", betas[3], ")", sep = ""))),
  y4 = eval(expression(paste("(", alphas[4], ", ", betas[4], ")", sep = ""))),
  y5 = eval(expression(paste("(", alphas[5], ", ", betas[5], ")", sep = "")))
)

pdfOrCdf_names <- c(
  cdf_y = "CDF",
  pdf_y = "PDF"
)

plot_labeller <- function(variable,value){
  if (variable=='tuple') {
    return(tuple_names[value])
  } else if (variable=='pdfOrCdf') {
    return(pdfOrCdf_names[value])
  } else {
    return(as.character(value))
  }
}

df <- cbind(pdf, cdf)
df <- df[-c(4, 5)]
names(df) <- c("x", "tuple", "PDF", "CDF")

df <- gather(df, pdfOrCdf, y, PDF, CDF)
df$pdfOrCdf <- factor(df$pdfOrCdf)
df <- subset(df, !(df$y >=5))


fig <- ggplot(df, aes(x, y, colour = tuple)) +
  geom_line(size = .5) +
  facet_grid(tuple ~ pdfOrCdf, scales = "free", labeller = plot_labeller) +
  ylab("y\n") +
  xlab("\nx") +
  theme(legend.position = "none") +
  theme(legend.title = element_text(size = 14),
        legend.text = element_text(size = 13),
        axis.text = element_text(size = 12),
        axis.title = element_text(size = 16),
        title = element_text(size = 18)) +
  theme(strip.text.y = element_text(size = 14),
        strip.text.x = element_text(size = 14)) +
  ggtitle(expression(paste( "PDF and CDF of Gamma function for several (",
                            alpha, ", ", beta, ") pairs", sep = "")))
```

```
## Warning: The labeller API has been updated. Labellers taking `variable`and
## `value` arguments are now deprecated. See labellers documentation.
```

```
#ggsave(plot = fig, filename = paste0("q2t3.pdf"), height = 8.5, width = 11)
print(fig)
```


PDF and CDF of Gamma function for several (α, β) pairs
CDF
PDF
(0.5, 0.5)
(5, 1)
(1, 3)
(2, 2)
(2, 5)
y
x
0
1
2
3
4
5
0.0
0.5
1.0
1.5
2.0
2.5
0.00
0.25
0.50
0.75
1.00

Question 3

The file `redfinData.txt` contains unstructured data about the housing market. Each row contains a monthly observation for either the national market or a specific market, e.g., San Francisco. We want to create a data frame where each of the variables is a column, along with year and region. The data in the data frame is be the mean for a given region/year combination.

```
redfin <- read.table("redfinData.csv", sep = "\t", fileEncoding = "UTF-16",
                     dec = ".", stringsAsFactors = F, header = T)
str(redfin, strict.width = "w")
```

```
## 'data.frame':    11628 obs. of  4 variables:
## $ Measure.Names : chr "Average Sale To List YoY " "Average Sale To List
##    YoY " "Average Sale To List YoY " "Average Sale To List YoY " ...
## $ Month.of.Period.End: chr "June 2016" "May 2016" "April 2016" "March
##    2016" ...
## $ Region : chr " National" " National" " National" " National" ...
## $ Measure.Values : num 0.00588 0.00551 0.00183 0.01074 0.00154 ...
```

We can see that the column `Measure.Names` contains many different variables that we want to put into separate columns. We want to convert this *tall* table into a *wide* table. First, we convert the `Month.of.Period.End` column to `Year` by removing the month from each entry.

```
# convert the column with Month+Year to only Year and rename it
redfin$Month.of.Period.End <-
  paste("01", redfin$Month.of.Period.End) %>%
    as.Date("%d %B %Y") %>%
      format("%Y")
names(redfin)[2] <- "Year"
```

Now we can convert the data frame into a tidier wide data frame, where the columns will be the desired variables. We check the result by taking a look at the structure of the new data frame `tidyRedfin`.

```
# Convert to a "wide" data frame with Year, Region, and all 18 Measures
tidyRedfin <- dcast(redfin, Year + Region ~ Measure.Names, mean)
```

```
## Using Measure.Values as value column: use value.var to override.
```

```
# Clean up the whitespace in some Measure names
names(tidyRedfin) <- sub("^ | $", "", names(tidyRedfin))
# Clean up the whitespace in some Region names
tidyRedfin$Region <- sub("^ | $", "", tidyRedfin$Region)
# Set the year to numeric
tidyRedfin$Year <- as.numeric(tidyRedfin$Year)

str(tidyRedfin, strict.width = "w")
```

```
## 'data.frame':    60 obs. of  20 variables:
## $ Year : num 2012 2012 2012 2012 2012 ...
## $ Region : chr "Boston, MA metro area" "Chicago, IL metro area" "Los
##    Angeles, CA metro area" "National" ...
## $ Average Sale To List : num 0.964 0.95 0.986 0.934 0.945 ...
## $ Average Sale To List MoM: num 0.001002 0.001348 0.001379 0.000858
##    0.000488 ...
## $ Average Sale To List YoY: num 0.006919 0.008554 0.007631 0.007663
##    0.000368 ...
## $ Days on Market : num 110.6 110.2 45.2 71.8 88.3 ...
## $ Days on Market MoM : num -1.83 -1.92 -3 -1.5 -4.17 ...
## $ Days on Market YoY : num -8.58 -12.17 -16.08 -12.75 -30.83 ...
## $ Homes Sold : num 4742 7935 6850 165908 2928 ...
## $ Homes Sold MoM : num 0.0313 0.0269 0.021 0.0152 0.0241 ...
## $ Homes Sold YoY : num 0.199 0.246 0.116 0.109 0.159 ...
## $ Inventory : num 27769 55077 17863 718120 20952 ...
## $ Inventory MoM : num -0.01511 -0.01719 -0.07391 -0.01881 -0.00754 ...
## $ Inventory YoY : num -0.0969 -0.1532 -0.4146 -0.1805 0.3602 ...
## $ Median Sale Price : num 297275 155042 327479 200350 210487 ...
## $ Median Sale Price MoM : num 0.00931 0.00428 0.0125 0.0121 0.00498 ...
## $ Median Sale Price YoY : num 0.03072 -0.01983 0.04603 0.07218 0.00205 ...
## $ New Listings : num 6139 11389 7348 190654 4518 ...
## $ New Listings MoM : num 0.0292 0.01992 -0.00581 0.00732 0.02245 ...
## $ New Listings YoY : num -0.00524 0.01815 -0.14781 -0.01311 0.38675 ...
```

As a result, our `tidyRedfin` data frame has 60 observations of 20 variables. There is information about 12 unique regions:

Boston, MA metro area, Chicago, IL metro area, Los Angeles, CA metro area, National, Philadelphia, PA metro area, San Francisco, CA - Excelsior, San Francisco, CA - Fairmount, San Francisco, CA - Forest Knolls, San Francisco, CA - Golden Gate Heights, San Francisco, CA - Haight Ashbury, Seattle, WA metro area, Washington, DC metro area,

for 5 years:

2012, 2013, 2014, 2015, 2016.