1. ZOMBIES!

News reports suggest that the impossible has become possible…zombies have appeared on the streets of the US! What should we do? The Centers for Disease Control and Prevention (CDC) zombie preparedness website (https://www.cdc.gov/cpr/zombie/index.htm) recommends storing water, food, medication, tools, sanitation items, clothing, essential documents, and first aid supplies. Thankfully, we are CDC analysts and are prepared, but it may be too late for others!

Our team decides to identify supplies that protect people and coordinate supply distribution. A few brave data collectors volunteer to check on 200 randomly selected adults who were alive before the zombies. We have recent data for the 200 on age and sex, how many are in their household, and their rural, suburban, or urban location. Our heroic volunteers visit each home and record zombie status and preparedness. Now it's our job to figure out which supplies are associated with safety!

In [2]:
```
# Read in the data
zombies <- read.csv("datasets/zombies.csv")

# Examine the data with summary()
summary(zombies)

# Create water-per-person
zombies$water.person <- zombies$water / zombies$household

# Examine the new variable
summary(zombies$water.person)
```

```
    zombieid         zombie         age             sex          rurality
 Min.   :  1.00   Human :121   Min.   :18.00   Female: 99   Rural   :98
 1st Qu.: 50.75   Zombie: 79   1st Qu.:29.00   Male  :101   Suburban:48
 Median :100.50                Median :42.00                Urban   :54
 Mean   :100.50                Mean   :44.41
 3rd Qu.:150.25                3rd Qu.:58.00
 Max.   :200.00                Max.   :85.00
   household        water            food            medication
 Min.   :1.00   Min.   : 0.00    Food    :110   Medication   : 94
 1st Qu.:2.00   1st Qu.: 0.00    No food: 90    No medication:106
 Median :2.50   Median : 8.00
 Mean   :2.68   Mean   : 8.75
 3rd Qu.:4.00   3rd Qu.: 8.00
 Max.   :6.00   Max.   :40.00
      tools                         firstaid             sanitation      clothing
 No tools:101   First aid supplies    :106   No sanitation:102   Clothing:126
 tools   : 99   No first aid supplies: 94   Sanitation   : 98   NA's    : 74




     documents
 Documents: 66
 NA's     :134
```

```
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.000   0.000   2.000   3.092   5.333  13.333
```

In [3]:
```
# These packages need to be loaded in the first @tests cell
library(testthat)
library(IRkernel.testthat)

# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

test_zombies <- read.csv("datasets/zombies.csv")
test_zombies$water.person <- test_zombies$water / test_zombies$household

run_tests(
    test_that("the dataset is correct", {
              expect_identical(zombies,
                               test_zombies,
                               info = "The data frame is not correct. Did you
 load it correctly and divide water by household?")
              })
)
```

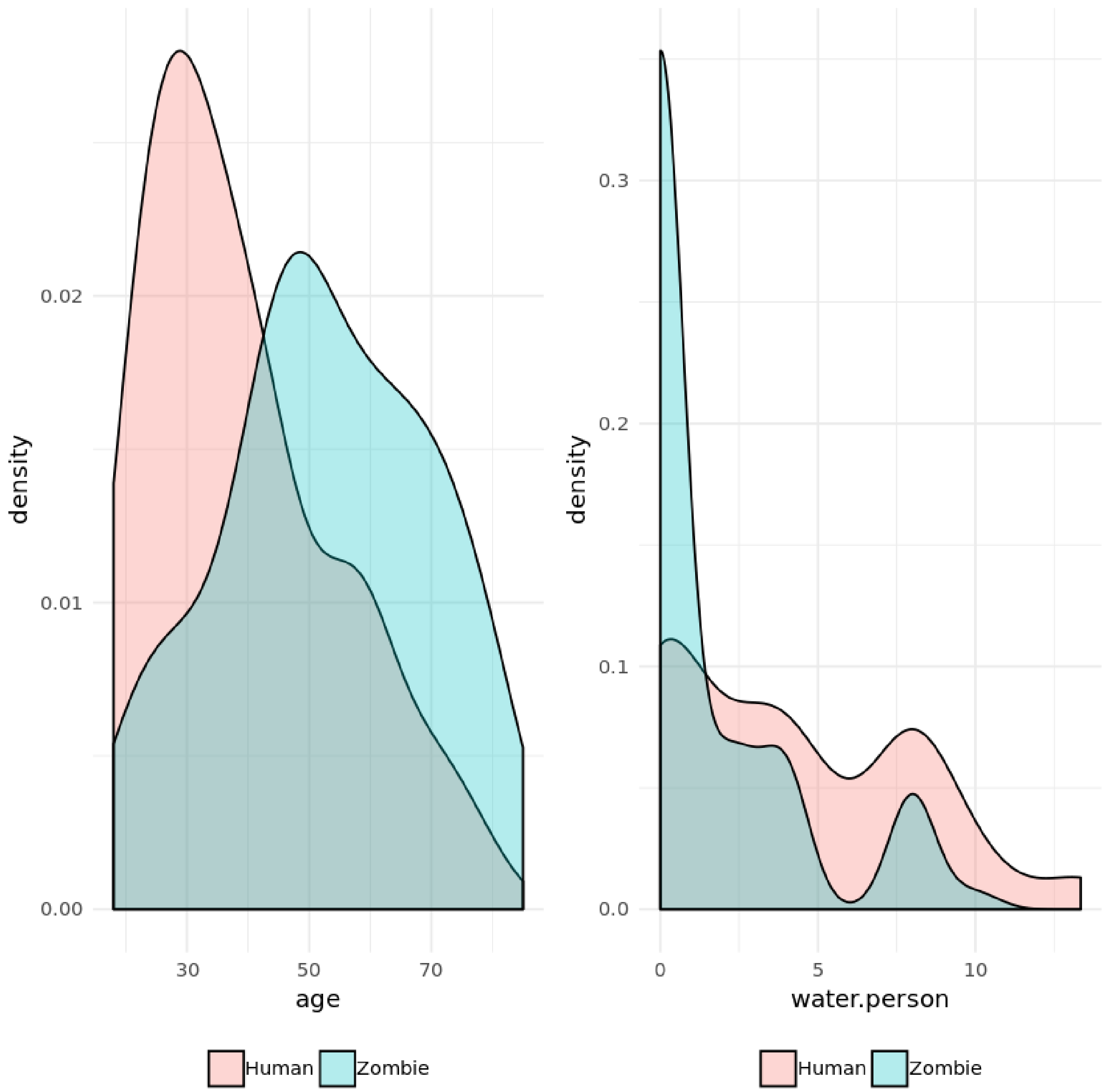
```
1/1 tests passed
```

2. Compare zombies and humans

Because every moment counts when dealing with life and (un)death, we want to get this right! The first task is to compare humans and zombies to identify differences in supplies. We review the data and find the following:

- zombieid: unique identifier
- zombie: human or zombie
- age: age in years
- sex: male or female
- rurality: rural, suburban, or urban
- household: number of people living in household
- water: gallons of clean water available
- food: food or no food
- medication: medication or no medication
- tools: tools or no tools
- firstaid: first aid or no first aid
- sanitation: sanitation or no sanitation
- clothing: clothing or no clothing
- documents: documents or no documents

In [4]:
```
# Load ggplot2 and gridExtra
library(ggplot2)
library(gridExtra)

# Create the ageZombies graph
ageZombies <- ggplot(data = zombies, aes(x = age, fill = zombie)) +
  geom_density(alpha = 0.3) +
  theme_minimal() +
  theme(legend.position = "bottom", legend.title = element_blank())

# Create the waterPersonZom graph
waterPersonZom <- ggplot(data = zombies, aes(x = water.person, fill = zombie))
+
  geom_density(alpha = 0.3) +
  theme_minimal() +
  theme(legend.position = "bottom", legend.title = element_blank())

# Display plots side by side
grid.arrange(ageZombies, waterPersonZom, ncol = 2)
```



In [5]:
```
# Create the test ageZombies graph
test_ageZombies <- ggplot(data = test_zombies, aes(x = age, fill = zombie)) +
  geom_density(alpha = 0.3) +
  theme_minimal() +
  theme(legend.position = "bottom", legend.title = element_blank())

# Create the test waterPersonZom graph
test_waterPersonZom <- ggplot(data = test_zombies, aes(x = water.person, fill
= zombie)) +
  geom_density(alpha = 0.3) +
  theme_minimal() +
  theme(legend.position = "bottom", legend.title = element_blank())

run_tests({
    test_that("packages are loaded", {
        expect_true("ggplot2" %in% .packages(), info = "Did you load the ggplo
t2 package?")
        expect_true("gridExtra" %in% .packages(), info = "Did you load the gri
dExtra package?")
})

    test_that("ageZombie is correct", {
        expect_identical(ageZombies$data,
                         test_ageZombies$data,
                         info = 'The data used in ageZombie is incorrect. Did
 you use zombies?')
        expect_identical(deparse(ageZombies$mapping$x),
                         deparse(test_ageZombies$mapping$x),
                         info = 'The x aesthetic in ageZombie is incorrect. Di
d you use age?')
        expect_identical(ageZombies$layers[[1]]$aes_params$alpha,
                         test_ageZombies$layers[[1]]$aes_params$alpha,
                         info = "alpha is incorrect. Please check its value.")
})

    test_that("waterPersonZom is correct", {
        expect_identical(waterPersonZom$data,
                         test_waterPersonZom$data,
                         info = 'The data used in waterPersonZom is incorrect.
Did you use zombies?')
        expect_identical(deparse(waterPersonZom$mapping$x),
                         deparse(test_waterPersonZom$mapping$x),
                         info = 'The x aesthetic in ageZombie is incorrect. Di
d you use water.person?')
        expect_identical(deparse(waterPersonZom$mapping$fill),
                         deparse(test_waterPersonZom$mapping$fill),
                         info = 'The fill aesthetic in ageZombie is incorrect.
Did you map it to zombie?')
        expect_identical(waterPersonZom$layers[[1]]$aes_params$alpha,
                         test_waterPersonZom$layers[[1]]$aes_params$alpha,
                         info = "alpha is incorrect. Please check its value.")
        expect_identical(waterPersonZom$theme,
                         test_waterPersonZom$theme,
                         info = "The theme is not correct. Please check it.")
    })
})
```

```
3/3 tests passed
```

3. Compare zombies and humans (part 2)

It looks like those who turned into zombies were older and had less available clean water. This suggests that getting water to the remaining humans might help protect them from the zombie hoards! Protecting older citizens is important, so we need to think about the best ways to reach this group. What are the other characteristics and supplies that differ between humans and zombies? Do zombies live in urban areas? Or are they more common in rural areas? Is water critical to staying human? Is food critical to staying human?

In [6]:
```
# Make a subset of the zombies data with only factors
zombies.factors <- zombies[ , sapply(zombies, is.factor)]

# Write a function to get percent zombies
perc.zombies <- lapply(zombies.factors,
                       function(x){
                           return(prop.table(table(x, zombies$zombie),
                                             margin = 1))
                           })
# Print the data
perc.zombies
```

```
$zombie

x        Human Zombie
  Human      1      0
  Zombie     0      1

$sex

x            Human    Zombie
  Female 0.6262626 0.3737374
  Male   0.5841584 0.4158416

$rurality

x              Human    Zombie
  Rural    0.8163265 0.1836735
  Suburban 0.5208333 0.4791667
  Urban    0.2962963 0.7037037

$food

x             Human    Zombie
  Food    0.8272727 0.1727273
  No food 0.3333333 0.6666667

$medication

x                   Human    Zombie
  Medication    0.8297872 0.1702128
  No medication 0.4056604 0.5943396

$tools

x              Human    Zombie
  No tools 0.6039604 0.3960396
  tools    0.6060606 0.3939394

$firstaid

x                            Human    Zombie
  First aid supplies    0.6320755 0.3679245
  No first aid supplies 0.5744681 0.4255319

$sanitation

x                   Human    Zombie
  No sanitation 0.4705882 0.5294118
  Sanitation    0.7448980 0.2551020

$clothing

x              Human    Zombie
  Clothing 0.5873016 0.4126984

$documents
```

```
x                Human    Zombie
  Documents 0.6666667 0.3333333
```

In [7]:

```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Make a subset of data with only factors
test_zombies.factors <- test_zombies[ , sapply(test_zombies, is.factor)]

# Write a function to get percent zombies
test_perc.zombies <- lapply(test_zombies.factors,
                       function(x){
                           return(prop.table(table(x, test_zombies.factors$zom
bie),
                                                      margin = 1))
                           })

run_tests({
    test_that("the subset is correct", {
        expect_identical(zombies.factors,
                         test_zombies.factors,
                         info = "The zombies.factors subset is incorrect. Did
 you supply the zombies data frame name to the sapply command?"
        )
    })

    test_that("the function is correct", {
        expect_equal(perc.zombies,
                         test_perc.zombies,
                         info = "The perc.zombies object is incorrect. Did you
provide the zombies.factors subset to the lapply command?"
        )
    })
})
```

```
2/2 tests passed
```

4. Recode variables missing values

Hmm…it seems a little fishy that the `clothing` and `documents` variables have only one category in `prop.table()`. After checking with the data collectors, they told you that they recorded those without clothing or documents as missing values or `NA` rather than `No clothing` or `No documents`.

To make sure the analyses are consistent and useful, the analysis team leader decides we should recode the `NA` values to `No clothing` and `No documents` for these two variables.

In [8]:
```
# Add new level and recode NA to "No clothing"
levels(zombies$clothing) <- c(levels(zombies$clothing), "No clothing")
zombies$clothing[is.na(zombies$clothing)] <- "No clothing"

# Add new level and recode NA to "No documents"
levels(zombies$documents) <- c(levels(zombies$documents), "No documents")
zombies$documents[is.na(zombies$documents)] <- "No documents"

# Check recoding
summary(
zombies)
```

```
    zombieid          zombie         age              sex          rurality
 Min.   :  1.00   Human :121   Min.   :18.00   Female: 99   Rural   :98
 1st Qu.: 50.75   Zombie: 79   1st Qu.:29.00   Male  :101   Suburban:48
 Median :100.50                Median :42.00                Urban   :54
 Mean   :100.50                Mean   :44.41
 3rd Qu.:150.25                3rd Qu.:58.00
 Max.   :200.00                Max.   :85.00
   household         water          food           medication
 Min.   :1.00   Min.   : 0.00   Food    :110   Medication   : 94
 1st Qu.:2.00   1st Qu.: 0.00   No food: 90    No medication:106
 Median :2.50   Median : 8.00
 Mean   :2.68   Mean   : 8.75
 3rd Qu.:4.00   3rd Qu.: 8.00
 Max.   :6.00   Max.   :40.00
       tools                     firstaid            sanitation
 No tools:101   First aid supplies   :106   No sanitation:102
 tools   : 99   No first aid supplies: 94   Sanitation   : 98



        clothing            documents     water.person
 Clothing   :126   Documents    : 66   Min.   : 0.000
 No clothing: 74   No documents:134    1st Qu.: 0.000
                                       Median : 2.000
                                       Mean   : 3.092
                                       3rd Qu.: 5.333
                                       Max.   :13.333
```

In [9]:
```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

levels(test_zombies$clothing) <- c(levels(test_zombies$clothing), "No clothin
g")
test_zombies$clothing[is.na(test_zombies$clothing)] <- "No clothing"

# Add new level and recode NA to "No documents"
levels(test_zombies$documents) <- c(levels(test_zombies$documents), "No docume
nts")
test_zombies$documents[is.na(test_zombies$documents)] <- "No documents"

run_tests({
    test_that("the clothing variable is correct", {
        expect_identical(zombies$clothing,
                         test_zombies$clothing,
                         info = "The clothing variable recoding is incorrect.
 Did you supply the right data frame and variable names?"
                        )
    })

    test_that("the documents variable is correct", {
        expect_identical(zombies$documents,
                         test_zombies$documents,
                         info = "The documents variable recoding is incorrect.
Did you supply the right data frame and variable names?"
                        )
    })
})
```

```
2/2 tests passed
```

5. Selecting variables to predict zombie status

From Task 3, it appears that 70.4% of people in urban areas are zombies, while just 18.4% of those in rural areas are zombies. Getting humans out of cities and protecting those who cannot leave seems important!

For most of the supplies, there is less of a difference between humans and zombies, so it is difficult to decide what else to do. Since there is just one chance to get it right and every minute counts, the analysis team decides to conduct bivariate statistical tests to gain a better understanding of which differences in percents are statistically significantly associated with being a human or a zombie.

In [10]:
```
# Update subset of factors
zombies.factors <- zombies[ , sapply(zombies, is.factor)]

# Chi-squared for factors
chi.zombies <- lapply(zombies.factors,
                      function(x){
                          return(chisq.test(x, zombies.factors$zombie))
                          })

# T-tests for numeric
ttest.age <- t.test(zombies$age ~ zombies$zombie)
ttest.water <- t.test(zombies$water.person ~ zombies$zombie)

# Examine the results
chi.zombies
ttest.age
ttest.water
```

```
$zombie

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 195.84, df = 1, p-value < 2.2e-16


$sex

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 0.21561, df = 1, p-value = 0.6424


$rurality

        Pearson's Chi-squared test

data:  x and zombies.factors$zombie
X-squared = 41.271, df = 2, p-value = 1.092e-09


$food

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 48.49, df = 1, p-value = 3.32e-12


$medication

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 35.747, df = 1, p-value = 2.247e-09


$tools

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 0, df = 1, p-value = 1


$firstaid

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 0.47178, df = 1, p-value = 0.4922


$sanitation
```

```
        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 14.61, df = 1, p-value = 0.0001322


$clothing

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 0.26864, df = 1, p-value = 0.6042


$documents

        Pearson's Chi-squared test with Yates' continuity correction

data:  x and zombies.factors$zombie
X-squared = 1.206, df = 1, p-value = 0.2721


        Welch Two Sample t-test

data:  zombies$age by zombies$zombie
t = -5.6247, df = 155.02, p-value = 8.453e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -18.099289  -8.690751
sample estimates:
 mean in group Human mean in group Zombie 
            39.12397             52.51899 

        Welch Two Sample t-test

data:  zombies$water.person by zombies$zombie
t = 5.5436, df = 197.43, p-value = 9.415e-08
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 1.636281 3.443253
sample estimates:
 mean in group Human mean in group Zombie 
            4.095041             1.555274 
```

In [11]:
```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Update subset of factors
test_zombies.factors <- test_zombies[ , sapply(test_zombies, is.factor)]

# Chi-squared for factors
test_chi.zombies <- lapply(test_zombies.factors,
                       function(x){
                           return(chisq.test(x, zombies.factors$zombie))
                           })

# T-tests for numeric
test_ttest.age <- t.test(test_zombies$age ~ test_zombies$zombie)
test_ttest.water <- t.test(test_zombies$water.person ~ test_zombies$zombie)

run_tests({
    test_that("the factor subset is correct", {
        expect_identical(zombies.factors,
                         test_zombies.factors,
                         info = "The zombies.factors subset is incorrect. Did
 you supply the right data frame?"
                        )
    })
    test_that("the chi squareds are correct", {
        expect_equal(chi.zombies,
                         test_chi.zombies,
                         info = "The chi-squared analyses are incorrect. \n Di
d you supply zombies.factors to both the lapply and chisq.test commands?"
                        )
    })
    test_that("the age t-test is correct", {
        expect_equal(ttest.age$statistic,
                         test_ttest.age$statistic,
                         info = "The t-statistic for the age t-test is incorre
ct. \n Did you put the age variable first and the zombie variable second in th
e command?"
                        )
    })
    test_that("the water t-test is correct", {
        expect_equal(ttest.water$statistic,
                         test_ttest.water$statistic,
                         info = "The t-statistic for the water t-test is incor
rect. \n Did you put the water variable first and the zombie variable second i
n the command?"
                        )
    })
})
```

```
4/4 tests passed
```

6. Build the model

Now we are getting somewhere! Rurality, food, medication, sanitation, age, and water per person have statistically significant relationships to zombie status. We use this information to coordinate the delivery of food and medication while we continue to examine the data!

The next step is to estimate a logistic regression model with `zombie` as the outcome. The generalized linear model command, `glm()`, can be used to determine whether and how each variable, and the set of variables together, contribute to predicting zombie status. Following `glm()`, `odds.n.ends()` computes model significance, fit, and odds ratios.

In [12]:
```
# Create zombie model
zombie.model <- glm(zombie ~ age + water.person + food + rurality + medication
+ sanitation,
                   data = zombies, family = binomial(logit))

# Model significance, fit, and odds ratios with 95% CI
library(odds.n.ends)
zombie.model.fit <- odds.n.ends(zombie.model)

# Print the results of the odds.n.ends command
zombie.model.fit
```

```
Waiting for profiling to be done...
```

$`Logistic regression model significance`

Chi-squared	145.596
d.f.	7
p	0

$`Contingency tables (model fit): percent predicted`

A table: 3 x 3 of type dbl

	1	0	Sum
1	0.315	0.060	0.375
0	0.080	0.545	0.625
Sum	0.395	0.605	1.000

$`Contingency tables (model fit): frequency predicted`

A table: 3 x 3 of type dbl

	1	0	Sum
1	63	12	75
0	16	109	125
Sum	79	121	200

$`Predictor odds ratios and 95% CI`

A matrix: 8 x 3 of type dbl

	OR	2.5 %	97.5 %
(Intercept)	0.00224594	0.0002093871	0.01622961
age	1.08005714	1.0485596858	1.11810998
water.person	0.78377398	0.6600894297	0.91287014
foodNo food	9.02618095	3.4071657576	26.70797145
ruralitySuburban	3.69686205	1.2545823697	11.59438611
ruralityUrban	14.55818400	4.5481528429	54.42513843
medicationNo medication	5.52134058	2.0232321555	16.53129848
sanitationSanitation	0.31417163	0.1177847153	0.78789714

$`Model sensitivity`

0.79746835443038

$`Model specificity`

0.900826446280992

In [13]:
```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Create zombie model
test_zombie.model <- glm(zombie ~ age + water.person + food + rurality + medic
ation + sanitation,
                   data = test_zombies, family = binomial(logit))

# Model significance, fit, and odds ratios with 95% CI
test_zombie.model.fit <- odds.n.ends(test_zombie.model)


run_tests({
    test_that("odds.n.ends is loaded", {
        expect_true("odds.n.ends" %in% .packages(), info = "Did you load the o
dds.n.ends package?")
})

    test_that("the model is correct", {
        expect_equal(zombie.model$coefficient[1],
                     test_zombie.model$coefficient[1],
                     info = "The intercept for zombie.model is incorrect. \n D
id you add the medication variable to the model? Is the data frame spelled cor
rectly as zombies?"
        )

    })
    test_that("the odds.n.ends output is correct", {
        expect_identical(zombie.model.fit,
                         test_zombie.model.fit,
                         info = "The odds.n.ends results are incorrect. \n Did
you enter the zombie.model object into the command?"
        )
    })
})
```

```
Waiting for profiling to be done...

3/3 tests passed
```

7. Checking model assumptions

The model is statistically significant ($\chi^2 = 145.6$; $p < 0.05$), indicating that the variables in the model work together to help explain zombie status. Older age, having no food, living in suburban or urban areas (compared to rural), and having no access to medication increased the odds of being a zombie. Access to sanitation and having enough water decreased the odds of being a zombie. The model correctly predicted the zombie status of 63 zombies and 109 humans, or 172 of the 200 participants. Before relying on the model, check model assumptions: no multicollinearity and linearity.

Checking multicollinearity:
We can use the generalized variance inflation factor (GVIF) to check for multicollinearity. The GVIF determines to what extent each independent variable can be explained by the rest of the independent variables. When an independent variable is well-explained by the other independent variables, the GVIF is high, indicating that the variable is redundant and should be dropped from the model. Values greater than two are often used to indicate a failed multicollinearity assumption.

$$GVIF^{(1/(2df))} < 2$$

df = degrees of freedom

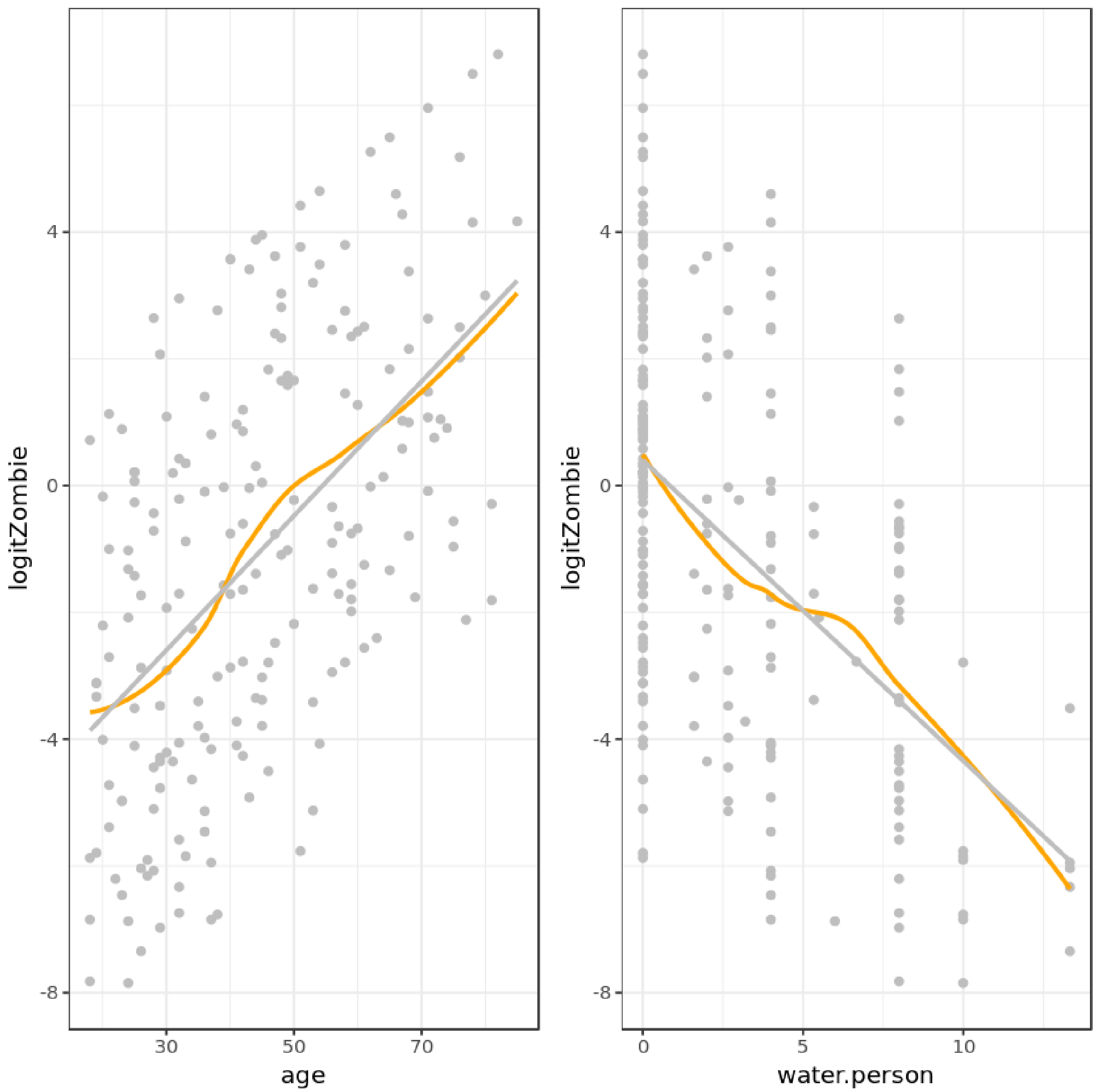
Checking linearity:
Linearity can be checked by graphing the log-odds of the outcome against each numeric predictor to see if the relationship is linear.

In [14]:
```
# Compute GVIF
library(car)
vif(zombie.model)

# Make a variable of the logit of the outcome
zombies$logitZombie <- log(zombie.model$fitted.values/(1-zombie.model$fitted.v
alues))

# Graph the logit variable against age and water.person
ageLinearity <- ggplot(data = zombies, aes(x = age, y = logitZombie))+
  geom_point(color = "gray") +
  geom_smooth(method = "loess", se = FALSE, color = "orange") +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  theme_bw()

waterPersonLin <- ggplot(data = zombies, aes(x = water.person, y = logitZombie
))+
  geom_point(color = "gray") +
  geom_smooth(method = "loess", se = FALSE, color = "orange") +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  theme_bw()

grid.arrange(ageLinearity, waterPersonLin, ncol = 2)
```

```
Loading required package: carData
```

A matrix: 6 x 3 of type dbl

	GVIF	Df	GVIF^(1/(2*Df))
age	1.508748	1	1.228311
water.person	1.188868	1	1.090352
food	1.304250	1	1.142038
rurality	1.313980	2	1.070649
medication	1.271348	1	1.127541
sanitation	1.102351	1	1.049929



In [15]:
```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Make a variable of the logit of the outcome
test_zombies$logitZombie <- log(test_zombie.model$fitted.values/(1-test_zombi
e.model$fitted.values))

# Graph the logit variable against age and water.person
test_ageLinearity <- ggplot(data = test_zombies, aes(x = age, y = logitZombie
))+
  geom_point(color = "gray") +
  geom_smooth(method = "loess", se = FALSE, color = "orange") +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  theme_bw()

test_waterPersonLin <- ggplot(data = test_zombies, aes(x = water.person, y = l
ogitZombie))+
  geom_point(color = "gray") +
  geom_smooth(method = "loess", se = FALSE, color = "orange") +
  geom_smooth(method = "lm", se = FALSE, color = "gray") +
  theme_bw()


run_tests({
    test_that("package car was loaded", {
        expect_true("car" %in% .packages(), info = "Did you load the car packa
ge?")
    })
    test_that("the logit variable is correct", {
        expect_identical(zombies$logitZombie,
                         test_zombies$logitZombie,
                         info = "The logitZombie is incorrect. Did you use zom
bie.model to compute the logit values?")
    })
    test_that("the age graph is correct", {
        expect_equal(ageLinearity$data,
                     test_ageLinearity$data,
                     info = "The data in ageLinearity are incorrect. Did you u
se zombies?")
        expect_identical(ageLinearity$mapping$x,
                         test_ageLinearity$mapping$x,
                         info = "The x aesthetic in ageLinearity is incorrect.
Did you use age?")
    })

    get_layers <- function(p) {
        unlist(c(list(p$layers), purrr::map(p$layers, "layers")))
    }

    test_that("the water graph is correct", {
        usr_lyrs <- get_layers(waterPersonLin)
        test_lyrs <- get_layers(test_waterPersonLin)

        expect_equal(waterPersonLin$data,
```

```
                     test_waterPersonLin$data,
                     info = "The data in waterPersonLin are incorrect. Did you
use zombies?")
        expect_identical(waterPersonLin$mapping$x,
                         test_waterPersonLin$mapping$x,
                         info = "The x aesthetic in waterPersonLin is incorrec
t. Did you use age?")
        expect_equal(usr_lyrs[[2]],
                     test_lyrs[[2]],
                     info = "The second geom layer is incorrect. \n Did you us
e geom_smooth() with method = 'loess' and se = FALSE?")
        expect_equal(usr_lyrs[[3]],
                     test_lyrs[[3]],
                     info = "The third geom layer is incorrect. \n Did you use
geom_smooth() with method = 'lm' and se = FALSE?")
    })
})
```

4/4 tests passed

8. Interpreting assumptions and making predictions

We find that the GVIF scores are low, indicating the model meets the assumption of no perfect multicollinearity. The plots show relatively minor deviation from the linearity assumption for `age` and `water.person`. The assumptions appear to be sufficiently met.

One of your friends on the analysis team hasn't been able to reach her dad or brother for hours, but she knows that they have food, medicine, and sanitation from an earlier phone conversation. Her 71-year-old dad lives alone in a suburban area and is excellent at preparedness; he has about five gallons of water. Her 40-year-old brother lives in an urban area and estimated three gallons of water per person. She decides to use the model to compute the probability they are zombies.

In [16]:
```
# Make a new data frame with the relatives data in it
newdata <- data.frame(age = c(71, 40),
                      water.person = c(5, 3),
                      food = c("Food", "Food"),
                      rurality = c("Suburban", "Urban"),
                      medication = c("Medication", "Medication"),
                      sanitation = c("Sanitation", "Sanitation"))

# Use the new data frame to predict
predictions <- predict(zombie.model, newdata, type = "response")

# Print the predicted probabilities
predictions
```

1	0.154576938664796
2	0.0972079734855113

In [17]:
```
# One or more tests of the student's code
# The @solution should pass the tests
# The purpose of the tests is to try to catch common errors and
# to give the student a hint on how to resolve these errors

# Make a new data frame with the relatives data in it
test_newdata <- data.frame(age = c(71, 40),
                      water.person = c(5, 3),
                      food = c("Food", "Food"),
                      rurality = c("Suburban", "Urban"),
                      medication = c("Medication", "Medication"),
                      sanitation = c("Sanitation", "Sanitation"))

# Use the new data frame to predict
test_predictions <- predict(test_zombie.model, test_newdata, type = "response"
)

run_tests({
    test_that("the data frame is accurate", {
        expect_identical(newdata,
                         test_newdata,
                         info = "The newdata data frame is incorrect. Did you
 fill in the correct lines with 40, 3, Food, Urban, Medication, and Sanitatio
n?"
                        )
    })
    test_that("the predicted probabilities are correct", {
        expect_equal(predictions,
                         test_predictions,
                         info = "The predicted probabilities are incorrect. Di
d you enter zombie.model into the predict command?"
                        )
    })
})
```

```
2/2 tests passed
```

9. What is your zombie probability?

Her dad has about a 15.5 percent chance of being a zombie and her brother has less than a 10 percent chance. It looks like they are probably safe, which is a big relief! She comes back to the team to start working on a plan to distribute food and common types of medication to keep others safe. The team discusses what it would take to start evacuating urban areas to get people to rural parts of the country where there is a lower percent of zombies. While the team is working on these plans, one thought keeps distracting you…your family may be safe, **but how safe are you?**

Add your own real-life data to the `newdata` data frame and predict your own probability of becoming a zombie!

In [18]:
```
# Add your data to the newdata data frame
newdata <- data.frame(age = c(71, 40, 25),
                      water.person = c(5, 3, 4),
                      food = c("Food", "Food", "Food"),
                      rurality = c("Suburban", "Urban", "Urban"),
                      medication = c("Medication", "Medication", "Medication"
),
                      sanitation = c("Sanitation", "Sanitation", "Sanitation"
))

# Use the new data frame to predict
predictions <- predict(zombie.model, newdata, type = "response")

# Print the predictions
predictions
```

1	0.154576938664796
2	0.0972079734855113
3	0.0258946448143426

In [19]:
```
test_newdata_length <- nrow(newdata)

run_tests({
    test_that("newdata has three observations", {
        expect_true(test_newdata_length == 3,
                    info = "The newdata data frame is incorrect. Did you fill
 in appropriate values for all the variables?"
                   )
    })
    test_that("age is numeric", {
        expect_true(is.numeric(newdata$age),
                    info = "The age variable must be a number."
                   )
    })
    test_that("water.person is numeric", {
        expect_true(is.numeric(newdata$water.person),
                    info = "The water.person variable must be a number."
                   )
    })
    test_that("food is correctly entered", {
        expect_true(newdata$food[3] %in% c("Food", "No food"),
                    info = "The food variable must Food or No food."
                   )
    })
    test_that("rurality is correctly entered", {
        expect_true(newdata$rurality[3] %in% c("Urban", "Suburban", "Rural"),
                    info = "The rurality variable value must be Urban, Suburba
n, or Rural."
                   )
    })
    test_that("medication is correctly entered", {
        expect_true(newdata$medication[3] %in% c("Medication", "No medication"
),
                    info = "The medication variable value must be Medication o
r No medication."
                   )
    })
     test_that("sanitation is correctly entered", {
         expect_true(
             newdata$sanitation[3] %in% c("Sanitation", "No sanitation"),
             info = "The sanitation variable value must be Sanitation or No sa
nitation."
         )
    })
})
```

7/7 tests passed

10. Are you ready for the zombie apocalypse?

While it is unlikely to be a zombie apocalypse will happen in the near future, the information presented in this notebook draws on emergency preparedness recommendations from the CDC. Although there is no way to make ourselves younger, we can have food, water, medication, and other supplies ready to ensure we are safe in the event of a blizzard, flood, tornado, or another emergency. After computing your zombie probability, think about what you could personally do to increase the likelihood that you will stay safe in the next storm or zombie apocalypse.

In [20]:
```
# What is your probability of becoming a zombie?
me <- 0.0258946448143426

# How prepared are you for a real emergency?
preparedness_level  <- "Okay, but I should probably pick up a few emergency it
ems at the store."
```

In [21]:
```
run_tests({
    test_that("me is numeric ", {
        expect_true(is.numeric(me),
                    info = "Did you assign your probability of becoming a zomb
ie to me? It should be a number."
                   )
    })
    test_that("preparedness_level is a string", {
        expect_true(is.character(preparedness_level),
                    info = "preparedness_level should be a character string."
                   )
    })
})
```

2/2 tests passed