

Actions on variables and rows

Melbourne Statistical Consulting Platform

University of Melbourne

April 2024

Data transformation in R

Actions on variables

- rename variables: `rename`
- select variables: `select`
- add new variables: `mutate`

Actions on rows

- select rows: `filter` and `slice`
- arrange rows: `arrange`

Data transformation in R

Actions on variables

- **rename variables:** `rename`
- **select variables:** `select`
- **add new variables:** `mutate`

Actions on rows

- select rows: `filter` and `slice`
- arrange rows: `arrange`

Renaming variables

```
screening_unclean <-  
  read_csv("../5 data/airport_screening_unclean.csv")
```

```
screening_unclean %>%  
  rename(arrival_port = port...7,  
         check_in_port = port...9,  
         period_of_stay = `period of stay`,  
         passenger_or_crew = P_C)
```

Motivation

- clarity (P_C?)
- consistency (arrival_port or ArrivalPort?)
- convenience ('Q4 I have trouble threading a needle for sewing clothes because of my poor eyesight')

Selecting variables

```
met <- read_csv("../5 data/MetManipulation.csv")
```

```
glimpse(met)
```

Rows: 11

Columns: 11

```
$ Department      <chr> "Drawings and Prints", "European Painting..."
$ Object_Name     <chr> "Petrarch's Laura", "A Hare and Birds", "...
$ Dynasty         <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
$ Reign           <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
$ Portfolio       <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA
$ Artist_Name     <chr> "Enea Vico", "Jan Fyt", "Pietro Testa", "...
$ Artist_Nationality <chr> "Italian", "Flemish", "Italian", "Dutch",...
$ Artist_Date     <dbl> 1523, 1611, 1612, 1616, 1741, 1838, 1886,...
$ Object_Date     <dbl> 1543, 1631, 1645, 1616, 1786, 1888, 1923,...
$ Object_Age      <dbl> 477, 389, 375, 404, 234, 132, 97, 106, 62...
$ Medium          <chr> "Engraving", "Oil on canvas", "Oil on can..."
```

Selecting variables

```
met <- read_csv("../5 data/MetManipulation.csv")
```

```
met %>%  
  select(-c(Dynasty, Reign, Portfolio))
```

Approach

- `select(Dynasty)` keeps Dynasty only
- `select(-Dynasty)` removes Dynasty only
- use `c()` to list more than one variable name to keep or remove
- `select(2:4, 8)` keeps the second, third, fourth and eighth columns
- `select(Reign:Medium)` keeps all columns from Reign to Medium inclusive
- `select(where(is.numeric))` keeps all the numeric variables
- use helper functions within `select()` to select based on conditions:
 - `starts_with("Artist")` selects variables that start with "Artist"
 - `ends_with("Date")` selects variables that end with "Date"
 - `contains("new")` selects variables that contain "new"

Mutating variables



Source: Allison Horst, 2018

`mutate()` function:

- creates new variables
- based on functions of existing variables
 - arithmetic operations
 - replacing levels
 - changing data types
 - ranks
 - and more

`mutate(..., .keep = 'none')`:

- the same but removes the old variables too
- previously called `transmute()`, but `transmute()` is now "superseded" by this usage of `mutate`

Mutating variables

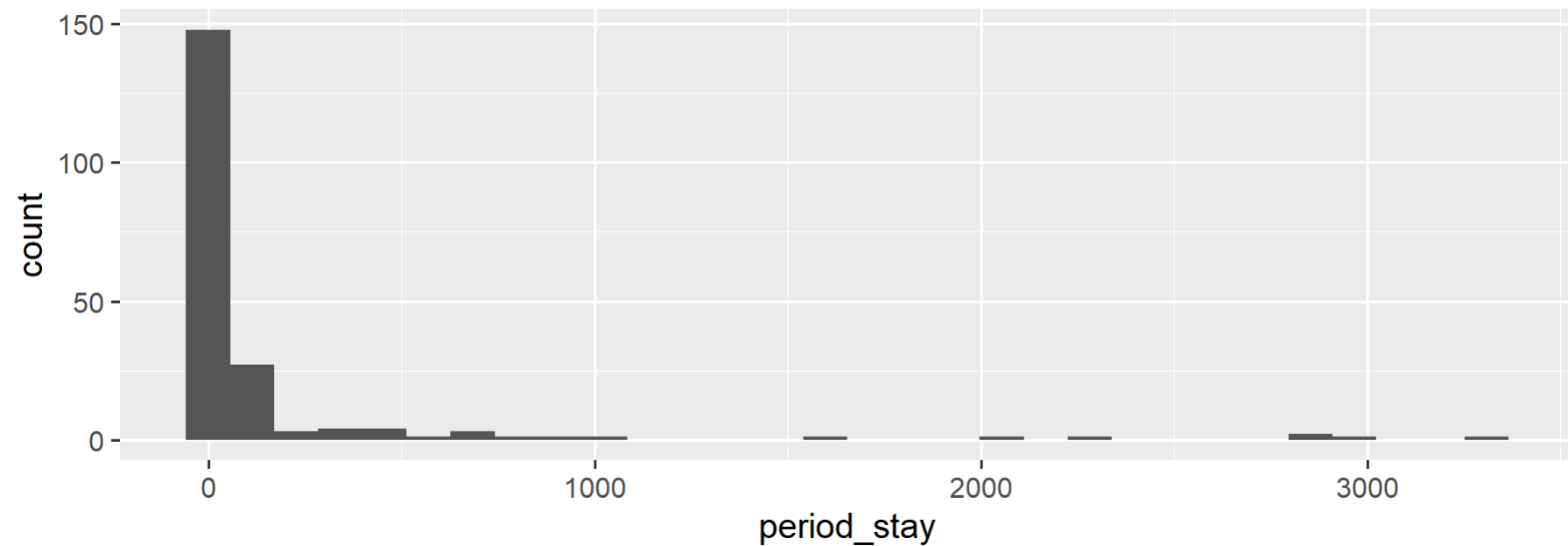
```
screening_unclean <-  
  read_csv("../5 data/airport_screening_unclean.csv")  
  
screening <- screening_unclean %>%  
  mutate(eggs = replace(eggs, eggs == "yes", 1)) %>%  
  mutate(eggs = as.numeric(eggs)) %>%  
  mutate(number_trips = na_if(number_trips, 9999))  
  
screening <- screening_unclean %>%  
  mutate(eggs = replace(eggs, eggs == "yes", 1),  
         eggs = as.numeric(eggs),  
         number_trips = na_if(number_trips, 9999))
```

In this example, we replaced the existing variables with these newly created variables, but it is not necessary to do so.

Mutating numerical variables

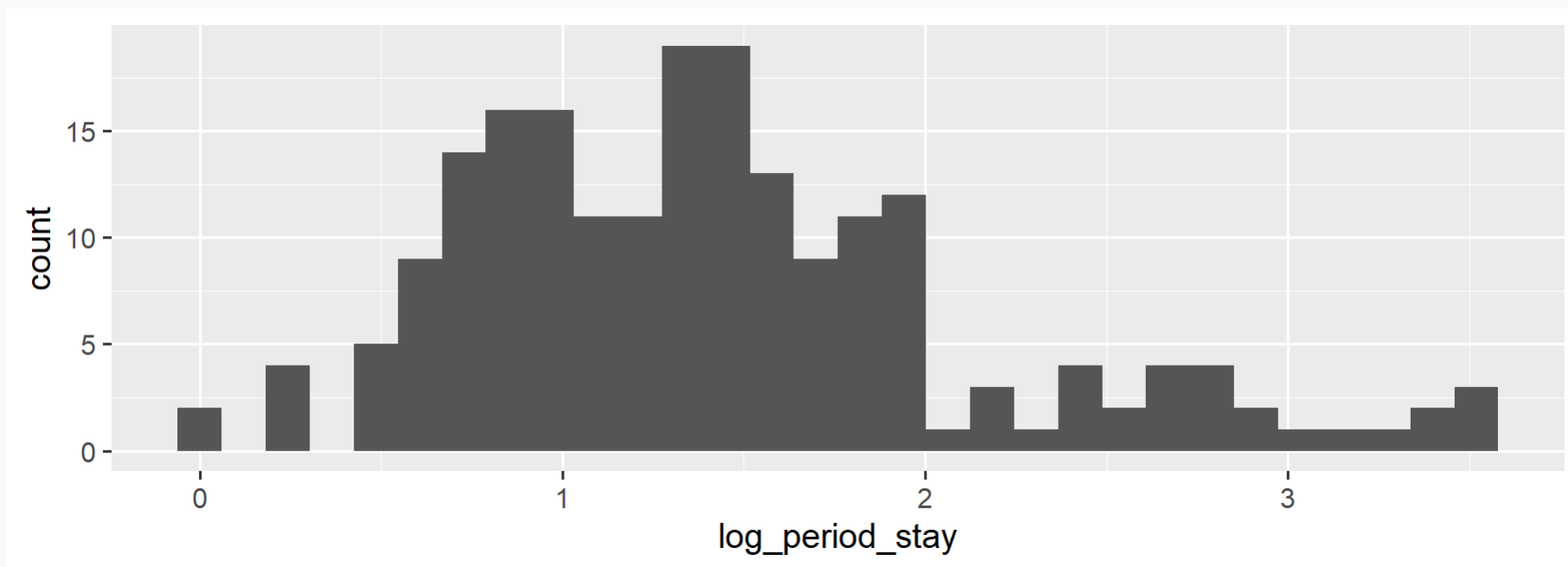
```
airport_screening <-  
  read_csv("../5 data/airport_screening.csv")
```

```
ggplot(airport_screening,  
      aes(x = period_stay)) +  
  geom_histogram(bins = 30)
```



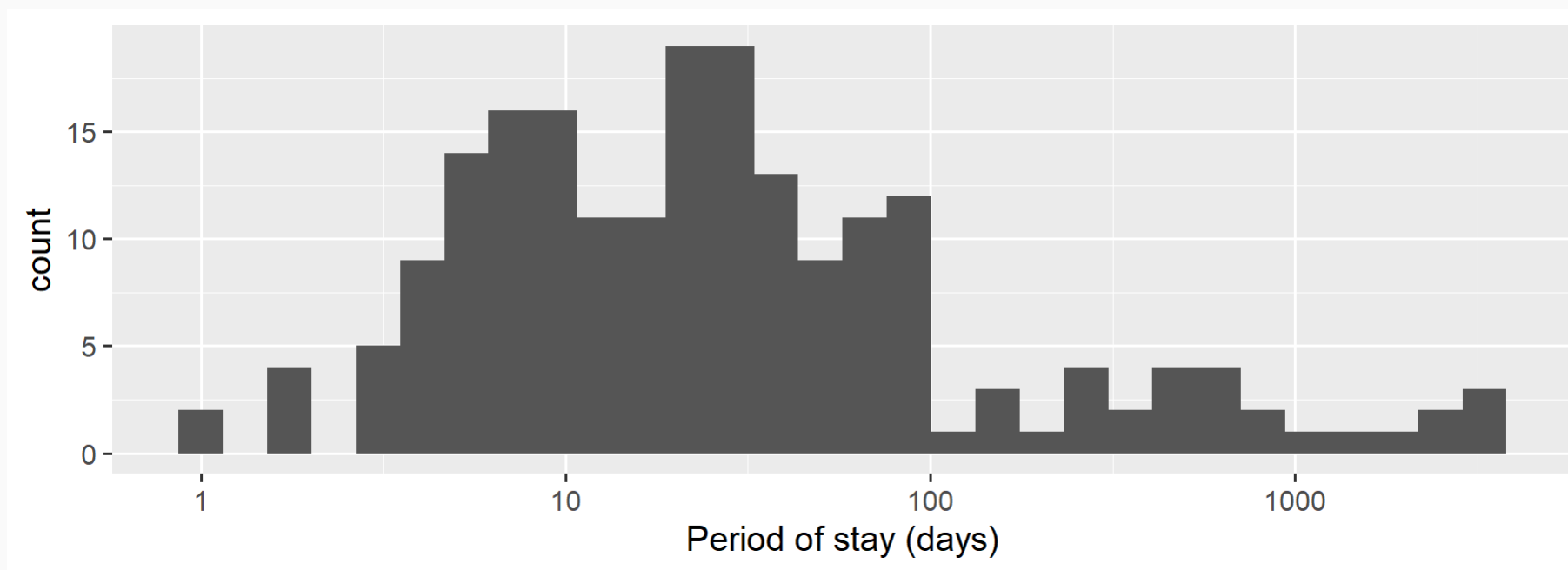
Mutating numerical variables

```
airport_screening <- airport_screening %>%  
  mutate(log_period_stay = log10(period_stay))  
  
ggplot(airport_screening, aes(x = log_period_stay)) +  
  geom_histogram(bins = 30)
```



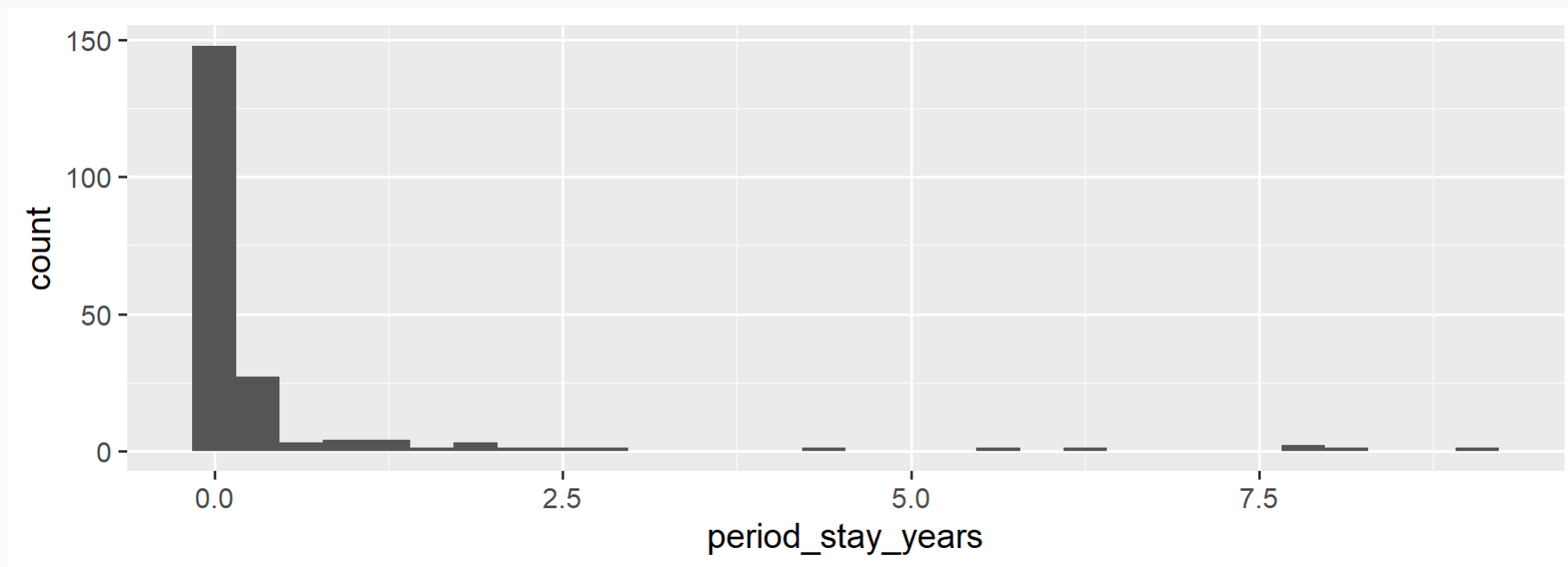
Mutating numerical variables

```
airport_screening <- airport_screening %>%  
  mutate(log_period_stay = log10(period_stay))  
  
ggplot(airport_screening, aes(x = log_period_stay)) +  
  geom_histogram(bins = 30) + labs(x = "Period of stay (days)") +  
  scale_x_continuous(breaks = 0:3, labels = c(1,10,100,1000))
```



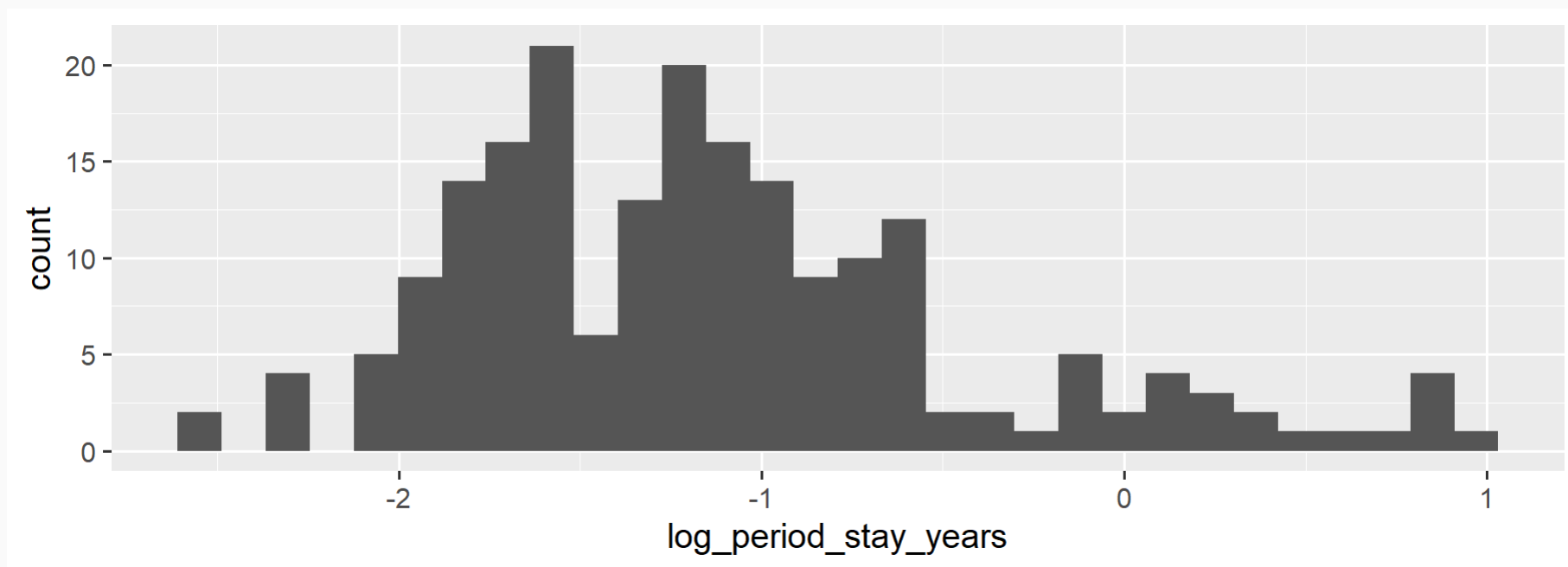
Mutating numerical variables

```
airport_screening <- airport_screening %>%  
  mutate(period_stay_years = period_stay/365)  
  
ggplot(airport_screening, aes(x = period_stay_years)) +  
  geom_histogram(bins = 30)
```



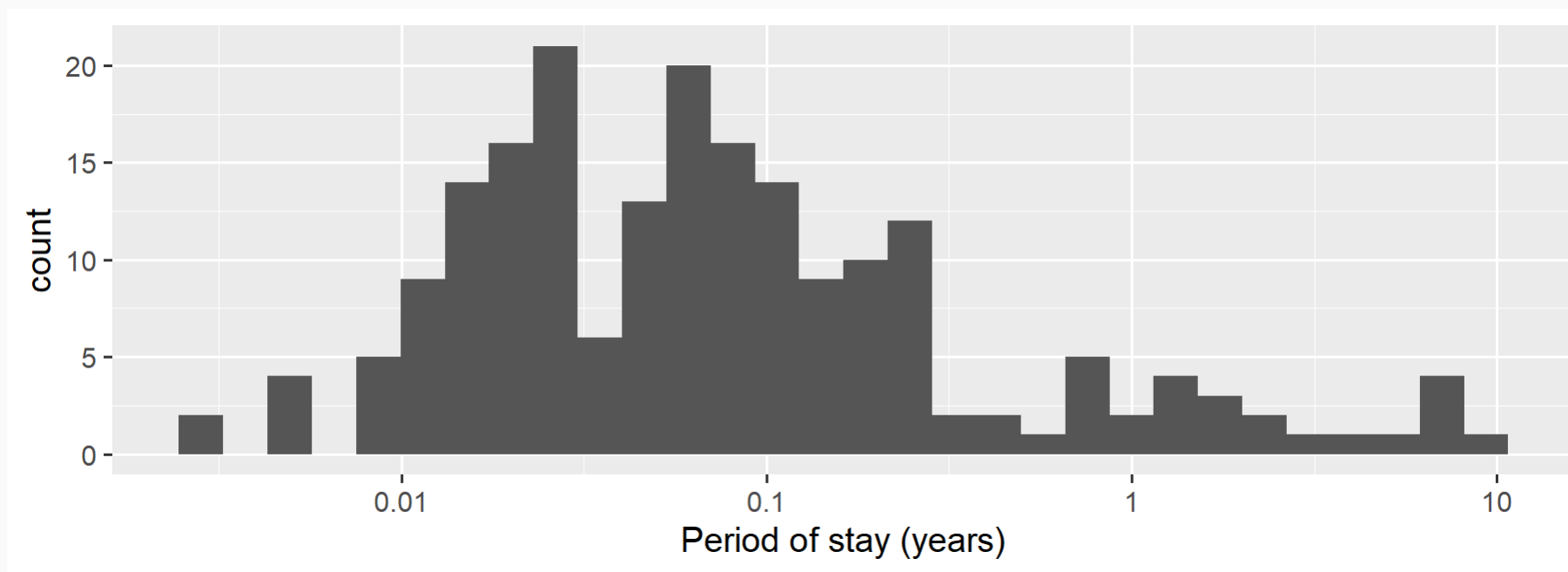
Mutating numerical variables

```
airport_screening <- airport_screening %>%  
  mutate(log_period_stay_years = log10(period_stay/365))  
  
ggplot(airport_screening, aes(x = log_period_stay_years)) +  
  geom_histogram(bins = 30)
```



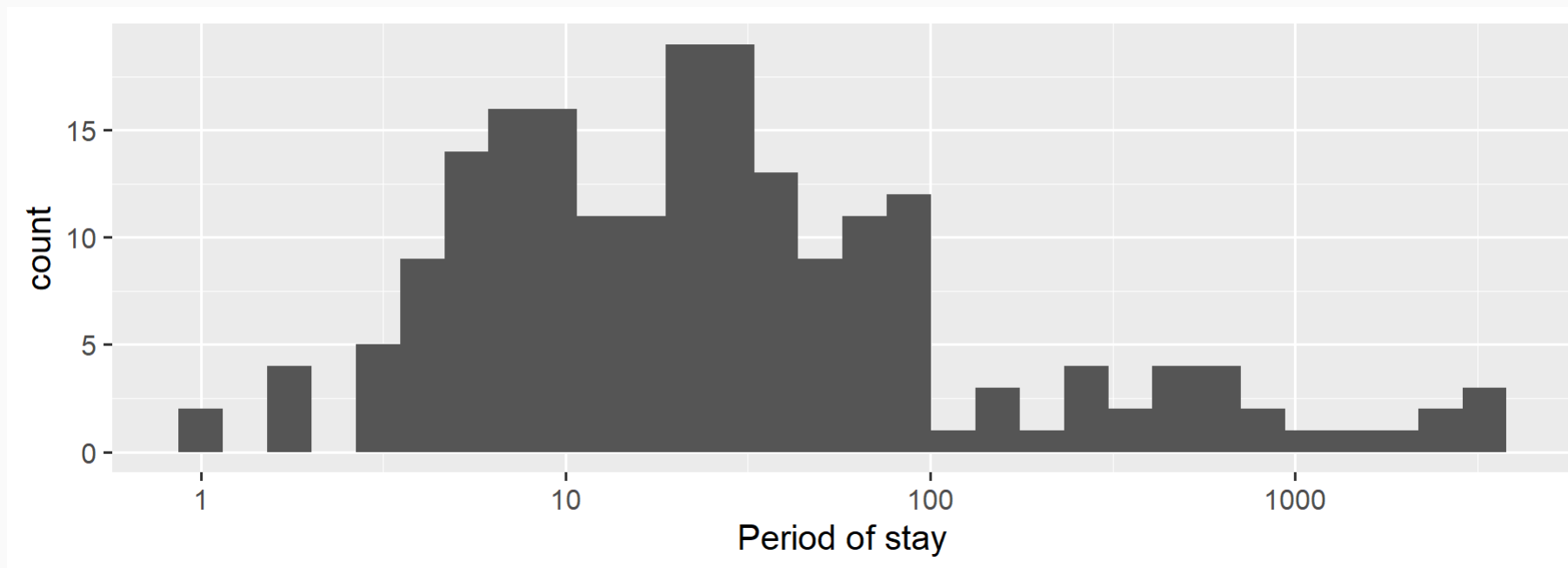
Mutating numerical variables

```
airport_screening <- airport_screening %>%  
  mutate(log_period_stay_years = log10(period_stay/365))  
  
ggplot(airport_screening, aes(x = log_period_stay_years)) +  
  geom_histogram(bins = 30) + labs(x = "Period of stay (years)") +  
  scale_x_continuous(breaks = -2:1, labels = c(0.01,0.1,1,10))
```



Mutating numerical variables

```
ggplot(airport_screening, aes(x = period_stay)) +  
  geom_histogram(bins = 30) + labs(x = "Period of stay") +  
  scale_x_continuous(trans='log10')
```



Data transformation in R

Actions on variables

- rename variables: `rename`
- select variables: `select`
- add new variables: `mutate`

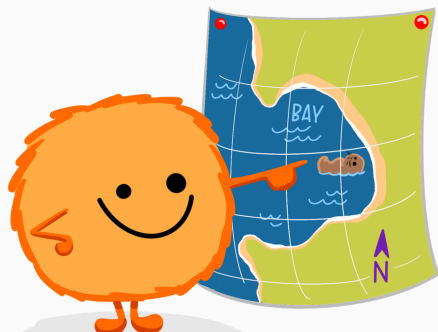
Actions on rows

- **select rows:** `filter` and `slice`
- **arrange rows:** `arrange`


Selecting rows

`dplyr::filter()` KEEP ROWS THAT
satisfy
your CONDITIONS

keep rows from... this data... ONLY IF... type is "otter" AND site is "bay"
`filter(df, type == "otter" & site == "bay")`



type	food	site
otter	urchin	bay
shark	seal	channel
otter	abalone	bay
otter	crab	wharf



The table shows data for different animal types, their food, and their location. The first, third, and fourth rows are highlighted in purple, indicating they are selected by the filter. The second row is highlighted in orange and marked with a red X, indicating it is excluded. The fourth row is also highlighted in orange and marked with a red X, indicating it is excluded. A purple character and a green character are standing next to the table, with a purple checkmark next to the first row and a green checkmark next to the third row.

Image by Allison Horst ([Twitter](#)).

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about passengers only?

```
airport_screening %>%  
  filter(passenger_crew == "P")
```

An introduction to Boolean operators

What happens if I want to select passengers only?

```
airport_screening %>%  
  filter(passenger_crew == "P")
```

What happens if I want to select passengers **AND** only those from Melbourne?

```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         arrival_port == "MEL" )
```

What happens if I want to select passengers **AND** either inexperienced travelers **OR** those arriving into Brisbane

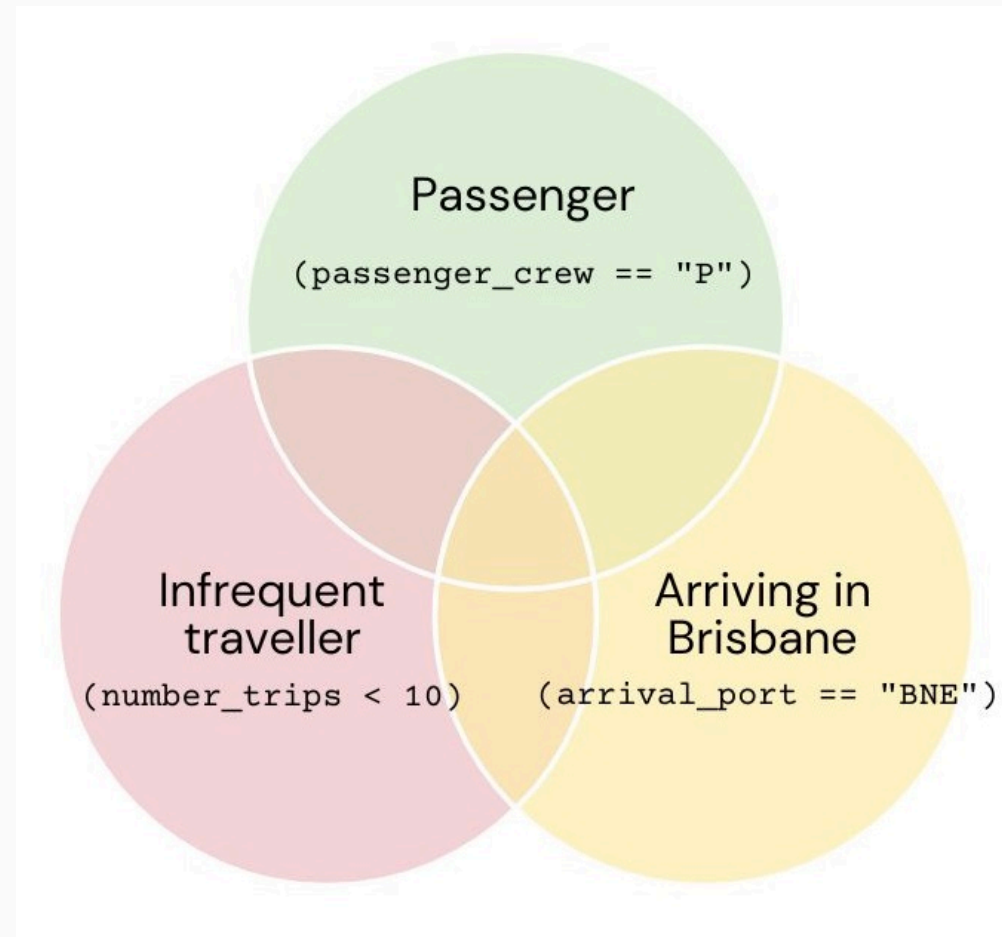
```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         (number_trips < 10 | arrival_port == "BNE"))
```

An introduction to Boolean operators

Operator	Operation
<code>x & y</code>	<code>x AND y</code>
<code>x y</code>	<code>x OR y</code>
<code>!x</code>	<code>not x</code>

Combinations of these operators can also be used, with careful parentheses

```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         (number_trips < 10 |  
          arrival_port == "BNE"))
```

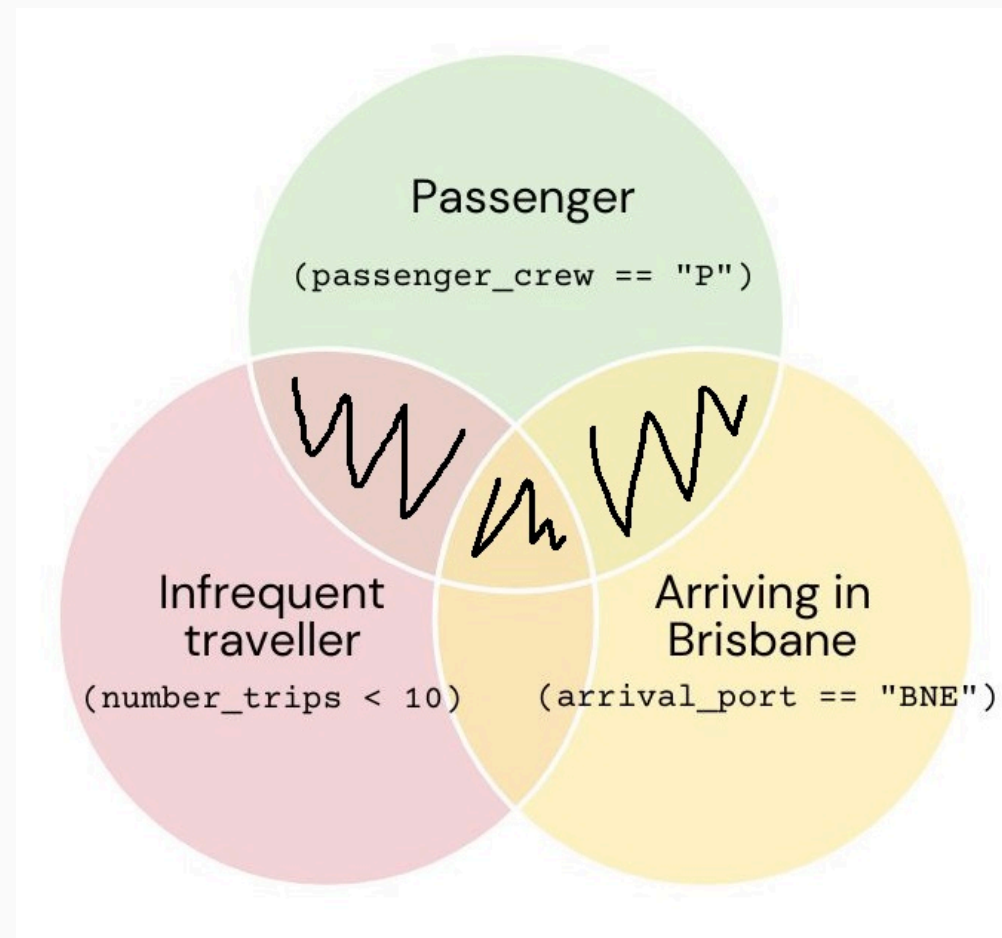


An introduction to Boolean operators

Operator	Operation
<code>x & y</code>	x AND y
<code>x y</code>	x OR y
<code>!x</code>	not x

Combinations of these operators can also be used, with careful parentheses

```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         (number_trips < 10 |  
          arrival_port == "BNE"))
```



Other useful operators in R

Operator	Operation
==	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
%in%	contained in

```
x <- c(5, 8, 10, 7, 14, 1, 14, 22)
x > 8
```

```
[1] FALSE FALSE TRUE FALSE TRUE FALSE TRUE TRUE
```

```
x >= 8
```

```
[1] FALSE TRUE TRUE FALSE TRUE FALSE TRUE TRUE
```

```
x == 14
```

```
[1] FALSE FALSE FALSE FALSE TRUE FALSE TRUE FALSE
```

```
x != 5
```

```
[1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

```
5 %in% x
```

```
[1] TRUE
```

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about passengers arriving in Melbourne only?

```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         arrival_port == "MEL") %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	54	88.5%
1	7	11.5%
Total	61	100.0%

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about passengers arriving in Melbourne or Sydney?

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about passengers arriving in Melbourne or Sydney?

```
airport_screening %>%  
  filter(passenger_crew == "P" &  
         arrival_port %in% c("MEL", "SYD")) %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	124	86.7%
1	19	13.3%
Total	143	100.0%

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about people who travel a lot?

Selecting rows

What percentage of people on incoming flights to Australia were found to have biosecurity risk material (BRM)?

Results for the full data frame

```
airport_screening %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	164	82.0%
1	36	18.0%
Total	200	100.0%

What about people who travel a lot?

```
airport_screening %>%  
  filter(number_trips > 10) %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	93	91.2%
1	9	8.8%
Total	102	100.0%

Selecting rows

We can also select rows by row number.

```
airport_screening %>%  
  slice(1:100) %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	80	80.0%
1	20	20.0%
Total	100	100.0%

```
airport_screening %>%  
  slice(1:20) %>%  
  tabyl(BRM) %>%  
  adorn_totals("row") %>%  
  adorn_pct_formatting() %>%  
  gt()
```

BRM	n	percent
0	17	85.0%
1	3	15.0%
Total	20	100.0%

But this is risky!

Arranging rows

Let's sort people screened to find the *least* frequent travellers and see which airports they are flying to and from and with which carriers

```
airport_screening %>%  
  arrange(number_trips) %>%  
  select(number_trips, check_in_port, arrival_port, airline, passenger_crew)
```

A tibble: 200 × 5

	number_trips	check_in_port	arrival_port	airline	passenger_crew
	<dbl>	<chr>	<chr>	<chr>	<chr>
1	2	NAN	BNE	FJ	P
2	2	SIN	MEL	QF	P
3	2	DXB	MEL	EK	P
4	2	WLG	SYD	QF	P
5	2	AKL	MEL	NZ	P
6	2	AKL	SYD	NZ	P
7	2	AKL	SYD	NZ	P
8	2	AKL	SYD	QF	P
9	2	DPS	PER	GA	P
10	2	BKK	SYD	QF	P

i 190 more rows

Arranging rows

Let's sort people screened to find the *most* frequent travellers and see which airports they are flying to and from and with which carriers (`desc` stands for descending order; ascending is the default)

```
airport_screening %>%  
  arrange(desc(number_trips)) %>%  
  select(number_trips, check_in_port, arrival_port, airline, passenger_crew)
```

A tibble: 200 × 5

	number_trips	check_in_port	arrival_port	airline	passenger_crew
	<dbl>	<chr>	<chr>	<chr>	<chr>
1	551	SIN	SYD	QF	C
2	479	PVG	SYD	QF	C
3	453	DFW	BNE	QF	C
4	437	DFW	BNE	QF	C
5	297	AKL	MEL	QF	P
6	228	BKK	SYD	BA	P
7	213	ICN	SYD	KE	C
8	175	NAN	MEL	FJ	P
9	171	HKG	SYD	VS	C
10	166	LAX	BNE	VA	P

i 190 more rows

Arranging rows

Let's sort people screened to find the *most* frequent travellers and see which airports they are flying to and from and with which carriers and focus just on passengers (alphabetically, C comes before P, so we want them in descending order)

```
airport_screening %>%  
  arrange(desc(passenger_crew), desc(number_trips)) %>%  
  select(number_trips, check_in_port, arrival_port, airline, passenger_crew)
```

A tibble: 200 × 5

	number_trips	check_in_port	arrival_port	airline	passenger_crew
	<dbl>	<chr>	<chr>	<chr>	<chr>
1	297	AKL	MEL	QF	P
2	228	BKK	SYD	BA	P
3	175	NAN	MEL	FJ	P
4	166	LAX	BNE	VA	P
5	134	SIN	ADL	SQ	P
6	124	POM	BNE	PX	P
7	116	AKL	SYD	EK	P
8	110	AKL	MEL	QF	P
9	100	NRT	SYD	QF	P
10	94	LHR	SYD	QF	P

i 190 more rows

Arranging rows

Let's sort people screened to find the *most* frequent travellers and see which airports they are flying to and from and with which carriers and focus just on passengers (which we can also do with filter, combining our two functions)

```
airport_screening %>%  
  filter(passenger_crew == "P") %>%  
  arrange(desc(number_trips)) %>%  
  select(number_trips, check_in_port, arrival_port, airline, passenger_crew)
```

A tibble: 192 × 5

	number_trips	check_in_port	arrival_port	airline	passenger_crew
	<dbl>	<chr>	<chr>	<chr>	<chr>
1	297	AKL	MEL	QF	P
2	228	BKK	SYD	BA	P
3	175	NAN	MEL	FJ	P
4	166	LAX	BNE	VA	P
5	134	SIN	ADL	SQ	P
6	124	POM	BNE	PX	P
7	116	AKL	SYD	EK	P
8	110	AKL	MEL	QF	P
9	100	NRT	SYD	QF	P
10	94	LHR	SYD	QF	P

i 182 more rows

Exercise 3.1.